# ArnLib

1.0.x

Generated by Doxygen 1.8.1

Mon Apr 1 2013 23:46:54

# Contents

# Chapter 1

# README



## ArnLib - Active Registry Network.

This Qt based library makes it easy to distribute changing data objects. It also gives a central place to find all your systems current data. By using the ArnBrowser, all data objects are real time presented in a tree view.

#### Comparison to similar concepts

- **Data mart:** Statistical data gathered from different systems. This makes it possible to run cross system analysis.
- **Windows Registry:** Centralized configuration data. All in one place easily shared.
- **ArnLib:** Hot changing data from different systems. Enables easy cross system data exchange, debugging, etc.

### Installation and usage

Read doc/Install.md how to build, install and use.

### Main features

- Based on QT, multiple platform and OS support.

#### Arn Data Objects

- Hierarchical storage of "hot" changing data objects.
- *Arn Data objects* can be: integers, floats, strings, byte arrays and variants (most QT data types, e.g. QImage).
- Data objects can typically be: measures, settings, datastreams, documents, scripts (js), ...
- *Arn Data objects* are thread-safe.
- Native support for data validation and double direction pipes.

### Sharing

- Data objects can be shared in a single program, among threads or between programs at different computers. This division of program modules can be changed and are transparent to usage of ArnLib.
- Support for temporary session data objects. Optional auto-delete of objects when tcp/ip closes and unique uuid names.
- Dependency system with custom offered services and getting signals when all needed services are available.
- Monitoring of newly created data objects and any mode change.

### Persistent storage

- Optional persistent storage of object in SQLight or in a file.
- Support for version control (VCS) of objects stored in files.

### Java Script

- Native support in JavaScript for: *Arn Data Objects*, Dependency system and Monitoring of changed objects.
- Java Script jobstack with preemptive and cooperative scripts running at different priorities.
- Hot swap of changed Java Script in jobstack.

### Data streams and *Remote Procedure Call*

- All data streams (pipes) can easily be monitored and manual test data can be inserted (see ArnBrowser).
- Service Api, for calling routines anywhere in connected Arn. *Remote Procedure call* (RPC) made simple as "remote signal slots".
- Service Api has an automatically generated help for giving syntax when doing debug manual typed calls to a RPC service.

# Chapter 2

# General Description

This document describes the general concepts of the ArnLib.

## 2.1 Arn Data Objects

All objects are stored in a tree hierarchy and the naming is similar to typical file systems, e.g. "//Measure/Water/-Temperature/value".

To get a handle to a folder, use a path ending with "/", e.g "//Measure/Water/".

Folder names can be empty. In the above example, the first level folder is empty and the second level folder is "Measure". The empty folder name can also be referred as "@". Again, the example can equally be written "/@/-Measure/Water/Temperature/value". This "@" is typically used when an empty name is unacceptable, e.g. in the tree viewer of ArnBrowser.

Each part in a given path is dynamically added as needed, i.e. any path can be used without explicitly creating each folder in advance.

### 2.1.1 Modes

*Mode* change is a one direction process. Once a specific *mode* is set, it can't be reset.

If the ArnItem is in a closed state when the *mode* change is done, the added modes will be stored and the real *mode* change is done when the ArnItem is opened to an *Arn Data Object*. This implies that ArnItems can benefit from *mode* settings before being opened.

If the *general mode* change is done to a `shared` object, the change of *general mode* is also done at the server and any connected clients.

The following *general modes* are available:

- **BiDir** A two-way object, typically for validation or pipe. See `bidirectional` objects.
- **Pipe** Implies *BiDir* and all data is preserved as a stream during `sharing`. Without *Pipe mode*, `sharing` is optimized to sync latest value and not all values in a stream.
- **Save** Sets the *Arn Data Object* as persistent and any data assigned to it will be saved. The persistent service must be started at the server. See `persistent` objects.

Additionally there are some *sync modes*. These modes are used by the local client session and are not shared with others. The *sync modes* must be set before the ArnItem is opened to an *Arn Data Object*.

Following *sync_modes* are available:

- **Master** The *Arn Data Object* (at client side) is set as *default generator* of data. Normally the server is the *default generator* of data. This makes difference when client connects or reconnects to the server. The data from the *default generator* is then used and synced.
- **AutoDestroy** The *Arn Data Object* (at client side) is set up for auto destruction. When the client closes tcp/ip, the server side will destroy the *Arn Data Object* and this will also be done at any connected clients.

Note: It's convenient to always set all the needed modes before an ArnItem is opened or an ArnItem is used as a template. See ArnItem::setTemplate().

### 2.1.2 Naming conventions

These rules must not be obeyed, but are recommended, to get the most benefits of the Arn echo system, like ArnBrowser.

- First level folder not empty, e.g "/MyLocalFolder/Key/value", is a local path and is not `shared`.
- First level folder empty, e.g. "//MyGlobalFolder/Date/value", is a global path and is `shared` to server and clients.
- When a leaf is used as an attribute, the following names are reserved:
  - **value** the value of the above closest folder denotation, e.g. "Temperature".
  - **set** allowed values and conversion to a more descriptive form, e.g. "0=Off 1=On".
  - **property** like precision and unit, e.g. "prec=1 unit=℃".
  - **info** like tool tips, e.g. "<tt>Standard UV radiation index</tt>".
  - **help.**XXX like "help.html" contains help in xhtml format.

## 2.2 Bidirectional Arn Data Objects

A bidirectional *Arn Data Object* is actually a double object, a twin. Each part has its own path but their life span is depending on each other.

One part is the normal "official" and the other part is *provider*. The provider has an added "!" to the normal path, e.g. normal = "//Measure/Depth/value", provider = "//Measure/Depth/value!".

Data written to one part ends up in the other. When a provider slot is connected to the provider part (ArnItem), the slot will receive "request" data from the normal part. The provider slot processes the request data and writes the result to the same provider part. This way the result will end up in the normal "official" part.

This functionality can typically be used for data validation and limiting.

### 2.2.1 Pipes

*Pipes* also use the `bidirectional` functionality. The two (twin) parts are then named *requester* and *provider*.

All data put into a pipe are part of a stream and as such will be fully transfered (syncronized) if they are `shared` with a server and other clients.

## 2.3 Persistent Arn Data Objects

The *server* must use ArnPersist to support the persistance service. As a standard, objects are stored in a SQLite database. It's also possible to store each object as a file.

Any connected *client* or the *server* can make an *Arn Data Object* persistent. It's just to open an ArnItem to the object and change *mode* to *Save*.

```
ArnItem arnMaxLevel;
arnMaxLevel.addMode( ArnItem::Mode::Save);
arnmaxLevel.open("//config/Level/Max/value");
```

When the *Arn Data Object* is set to *Save* mode, it's automatically loaded by the ArnPersist. At the *server* this is instantly done. A *client* has to wait for the value to get synced from the *server*. It's convenient to use ArnDepend to get a signal when the value is loaded and ready to use.

When the *Arn Data Object* is changed, it will be automatically saved by ArnPersist. There is a delay from first change of the object until the saving is done, see ArnItem::setDelay(). This allows for intensive updates of the object without choking down the server with saving operations.

It's possible to mark an object in the SQLite data base as *mandatory*. In this way the *Arn Data Object* is set as *persistent* and gets loaded at start of ArnPersist.

### 2.3.1 Saving objects in files

To use the *persistent* storing of *Arn Data Objects* in files, the *root* directory is set by: ArnPersist::setPersistDir(). This can also be combined with support of VCS (version control system).See ArnPersist::setVcs(). Currently there is a support module for *git*.

In the *root* directory and below, all (VCS) persistent files are stored. The *root* directory corresponds to the *root* in Arn tree.

Example: *root* directory is set to "/usr/local/arn_persist". There > is a file stored

at "/usr/local/arn_persist/@/doc/help.html". This > file will be mapped to Arn at "//doc/help.html".

Any files stored in the *root* directory and below, get loaded into their *Arn Data Object* with *mode* set as *persistent* at start of ArnPersist.

The files get updated in a similar way to the data base update.

## 2.4 Sharing Arn Data Objects

A fundamental aspect of *Arn* is that *Arn Data Objects* can be shared. This is centralized to the *Arn Server*, which stores all shared objects. It's still a distributed model as each client and server has their own set of *Arn Data Objects* that operate independent of any connection.

Each *Arn Client* connects to the *Arn Server* and decides which part of the *Arn Data Object* tree to be shared.

`ArnClient::setMountPoint("/share/")` will make the tree"/share/" shared.

This doesn't mean that everything in the shared tree at the server now will be available at the client. The client has to create an *Arn Data Object* in the shared tree. The client can then decide the exact objects of interest.

`ArnItem::Open("/share/Test/value")` will open a shared object (in previous example).

Note: Normally "//" is used for global (shared). See `naming conventions`.

## 2.5 RPC and SAPI

ArnRpc is the basic functionality of RPC (Remote Procedure Call). ArnSapi implements SAPI (Service Application Programming Interface) and is using ArnRpc as its base. It's recommended to use ArnSapi which has a higher level model.

The SAPI works by a model which can be described as RPC by *remote signal slots*. The *provider* is usually assumed to wait for a *requester* to initiate the session and then react to different remote calls from the *requester*. However, this is full duplex, so any side can make a remote call at any time.

A good example of the usage of SAPI is the *Arn Demo Chat*, which is included in the source package of the *ArnLib*.

ArnRpc uses pipes to communicate. The *pipes* can be monitored and receive test stimuli from the *Arn Browser* program. The used `protocol` is XString based and quite easy to handtype when common data types are used. "$help" will give the syntax for the actual custom SAPI.

A SAPI is setup by deriving the ArnSapi class to a new class that defines the *custom SAPI*. This custom-declared class is included at both the *provider* and *requester* ends. The *custom SAPI* class by itself doesn't implement any *services*. It's merely a hub for connections to *external signals and slots*. The base ArnSapi class automatically transfers all *custom signal* (SAPI) calls to the remote connected ends, which also have the ArnSapi derived class and that emits the transfered signal.

The provider connects the signals from custom SAPI that are prefixed with "pv_" (as default) to each external slot that implements the services. In the same way the *requester* connects the signals prefixed with "rq_" to its external "service" slots.

When there is a naming pattern between the *SAPI services* and the *external signals and slots*, it's a great convenience to use ArnRpc::BatchConnect(). This saves a lot of QObject::connect() calls. Also newly added services in the SAPI, that obey the naming scheme, will automaically be connected to the newly matching *external signals and slots* for implementation of the *service*.

An extended feature comparing to normal *signals* is that the *SAPI signals* are *public* and can be called by non-derived classes. This makes it optional to use both *signal to signal* connections or direct *signal* calls when issuing a RPC to the remote side.

The *service* slot can get the emitting *custom SAPI* object by using normal QObject::sender() functionality.

### 2.5.1 RPC and SAPI communication format

The RPC calling has a basic format as XString (see XStringMap). The most generic form is seen below. The type mark *T* is "t" for writeable types and "tb" for binary (non writeable) types.

> *funcname T*=_type1_ a._label1_=_arg1_ *T*=_type2_ a._label2_=_arg2_ ...
> Example: put t=QString a.id=level t=int a.value=123
> For calling: put( QString("level"), 123)

Commonly used *types* have a shorter form. The *types* are:

> *int*, *uint*, *bool*, _ba (QByteArray), list (QStringList), and default is QString

This can be used in previous example:

> put a.id=level int.value=123

Or even shorter, skipping labels, when typed by hand:

> put level int=123

List (QStringList) can be used. The *le* is *list element*. All examples below will get same resulting call.

> For a function: void test( QStringList lst, int num)
> test list=red green blu int=3
> test list.lst=red green blu int.num=3
> test list= le=red le=green le=blu int=3
> test list=red le=green blu int=3

For special cases, like empty elements, the *le* (list element) is needed. The example below has a first empty element followed by "green".

> test list= le= green blue int=2

The built-in call "$help" will give an automatically generated list of the present SAPI with the syntax for each available service.

## 2.6 Application notations

- If any graphics are used, Gui must be included.
- If only using QImage, Windowing system can be off, like: QApplication a(argc, argv, false);

# Chapter 3

# Installation and usage

## 3.1 Introduction

This software uses qmake to build all its components. qmake is part of a Qt distribution.

qmake reads project files, that contain the options and rules how to build a certain project. A project file ends with the suffix "$*$.pro". Files that end with the suffix "$*$.pri" are included by the project files and contain definitions, that are common for several project files.

The first step is to edit the $*$.pri / $*$.pro files to adjust them to your needs. Take care to select your deployment directories.

## 3.2 Documentation

The documentation is built by:

```
qmake
make doc
```

ArnLib includes a class documentation, that is available in various formats:

- **Html files**
- **PDF document**

  refman.pdf is built by:

  ```
  cd doc/latex
  make
  ```

- **Qt Compressed Help** ($*$.qch ) for the Qt assistant or creator.

  Load the doc/qthelp/arnlib.qch file into Qt Creator. Start Qt creator and go to Tools $>$ Options, open up Help and Documentation. Click Add and browse for the qch file that was just created, then Apply. It's best to close Qt creator at this point, and restart it.

## 3.3 Building ArnLib

The software can be built both by command line and IDE (Qt Creator). When using IDE, don't forget the "make install" step.

### A) Unix

```
qmake
make
make install
```

The easiest way of installing this library, is to let it be placed in a standard location for librarys and includes, e.g. /usr/lib and /usr/include/ArnLib. When using a shared library it's path has to be known to the run-time linker of your operating system. On Linux systems read "man ldconfig" ( or google for it ). Another option is to use the LD_-LIBRARY_PATH (on some systems LIBPATH is used instead, on MacOSX it is called DYLD_LIBRARY_PATH) environment variable.

If you only want to check the library examples without installing something, you can set the LD_LIBRARY_PATH to the lib directory of your local build.

The examples is built this way:

```
cd examples/ArnDemoChat
qmake
make
```

### B) Win32/MSVC

Has not been tested yet ...

Check that your Qt version has been built with MSVC - not with MinGW !

Please read the qmake documentation how to convert your $*$.pro files into your development environment.

For example MSVC with nmake:

```
qmake ArnLib.pro
nmake
nmake install
```

The examples is built this way:

```
cd examples
qmake ArnDemoChat.pro
nmake
```

Windows doesn't like mixing of debug and release binaries.

In windows it's possible to install the dll files together with the application binary, as the application directory always is included in the search path for dll.

### C) Win32/MinGW

Using Qt Creator for windows, will give you the needed tools for building a Qt project.

Check that your Qt version has been built with MinGW - not with MSVC !

Start a Shell, where Qt4 is initialized. (e.g. with "Programs->Qt by Trolltech ...->Qt 4.x.x Command Prompt" ). Check if you can execute "make" or something like "mingw32-make".

```
qmake ArnLib.pro
make
make install
```

The examples is built this way:

```
cd examples
qmake ArnDemoChat.pro
make
```

Windows doesn't like mixing of debug and release binaries.

In windows it's possible to install the dll files together with the application binary, as the application directory always is included in the search path for dll.

#### D) MacOSX

Has not been tested yet ...

Well, the Mac is only another Unix system. So read the instructions in A).

In the recent Qt4 releases the default target of qmake is to generate XCode project files instead of makefiles. So you might need to do the following:

```
qmake -spec macx-g++
```

#### E) Qt Embedded

ArnLib has been built with Qt Embedded using a Raspberry Pi. To build was as simple as for a regular Unix build.

## 3.4 Using ArnLib

In the ∗.pro file of the application the follwing can be used:

```
win32:CONFIG(release, debug|release): LIBS += L$$OUT_PWD/../ArnLib/release/ -lArn
else:win32:CONFIG(debug, debug|release): LIBS += -L$$OUT_PWD/../ArnLib/debug/ -lArn
else:unix: LIBS += -L$$OUT_PWD/../ArnLib/ -lArn
INCLUDEPATH += $$PWD/..
```

This will give a starting point for the configuration. It works well when using the same base directory for ArnLib as the application, e.g. basedir/ArnLib and basedir/myApp. In Unix alike systems it's also needed to install the library files in a path known by the system, see a) Unix.

If you don't use qmake you have to add the include path to find the ArnLib headers to your compiler flags and the ArnLib library to your linker list.

```
This Install.md file is based on documentation in the Qwt project.
```

# Chapter 4

# ArnLib Internals

This document describes internal processes that are relatively complex and by this needs some explanation.

## 4.1 ScriptJobs

- Each jobstack ScriptJobs is setup with a ScriptJobFactory wich makes custom interfaces etc.
- ScriptJobControl is setup with: Sriptfile, Config (QObject) and InterfaceList. Scriptfile is also copied to a ArnItem.
- ScriptJobControl can be connected to update of script in Arn, to make reload possible.
- Error text from ScriptJobControl can be connected to a pipe in Arn for logging.
- ScriptJobControl together with jobpriority define the ScriptJob and is added to ScriptJobs. Error text from Script job is connected to ScriptJobControl.
- Starting ScriptJobs in cooperative mode:
  1. Every ScriptJob is created and setup by corresponding ScriptJobControl
  2. Every ScriptJob is connected to Scheduler (yield etc).
  3. Every ScriptJobControl is connected to ScriptJobs for signaling update of script.
  4. Scheduler is started.
- Setup ScriptJob by ScriptJobControl:
  1. set ScriptJobFactory and Config
  2. Make and add the jobs Interfaces
  3. Evaluate the script (in js engine)
  4. run script function jobInit()
- Updating Script in cooperative mode:
  1. ScriptJobControl gets updated by Arn (or other).
  2. ScriptJobControl sends signal to ScriptJobs, which sets an updated flag for the corresponding Script Job.
  3. When scheduling, every updated script will get its sigQuit signal invoked and then reloaded.
  4. Reloading includes creating a new ScriptJob and setting up with ScriptJobControl etc.

- Starting ScriptJobs in preemtive mode:

  1. Every ScriptJob gets its own thread which also is setup with ScriptJobControl and ScriptJobFactory.
  2. Thread is started and it create a ScriptJobSingle where followning steps are done.
  3. ScriptJob is created and setup by ScriptJobControl
  4. ScriptJob is connected to Scheduler (yield etc).
  5. ScriptJobControl is connected to ScriptJobSingle for signaling update of script.
  6. Scheduler is started in ScriptJobSingle (just one job).

- Updating Script in preemtive mode:

  1. ScriptJobControl gets updated by Arn (or other).
  2. ScriptJobControl sends signal to ScriptJobSingle, which sets an updated flag and both invokes sigQuit signal to script and calls quit in scriptJob.
  3. ScriptJob aborts its js script engine and posts a custom Quit event with high prio.
  4. When ScriptJob get the Quit event, it will send a QuitRequest signal to ScriptJobSingle.
  5. ScriptJobSingle will get the signal amd detect update flag, which means reloading.
  6. Reloading includes creating a new ScriptJob and setting up with ScriptJobControl etc.

## 4.2 ArnMonitor

- Monitor starts its actual connection job when monitorPath is set.

- Monitor (at client-side) creates an ItemNet with path to monitorPath.

- The ItemNet is also put in syncQueue (always main-thread).

- Monitor puts the arn-event "monitorStart" in event loop, which makes sure event is sent after Monitor (and its caller) has finished initiializing.

- When "monitorStart" is received on local (client) side, the ItemNet will change SyncMode to Monitor. This will resync ItemNet to a Monitor at any server restart.

- Now 2 possibilities depending on threading:

  1. The ItemNet was sent before syncMode Monitor was set. Then server will receive an ordinary Itemnet and do standard setup.
  2. The ItemNet was sent with syncMode Monitor set. The server will detect this and do MonitorSetup on the ItemNet.

- When arn-event "monitorStart" is received on server-side, if SyncMode is not already set to "Monitor", server will do MonitorSetup on the ItemNet.

- When doing MonitorSetup (at server-side), connections are made to send arn-events when new childs are created, and present childs are directly sent as arn-event.

## 4.3 Destroy

- Command arives with a netId.
- Corresponding ItemNet is disabled (set as defunct).
- All link-leaves for the ItemNet:s tree is set as retired and each leaf is emitting a retired signal.
- The retired signal is handled by each connected Item. Each Item is sending a linkDestroyed signal to be handled by application code. The Items is finally closed and by this the link ref counter is decremented.
- When the links ref counter is reaching zero, a zeroRef signal is sent.
- The signal is handled by doZerRefLink(), in Main thread. It will set the link ref counter to -1 to mark the link as fully de-referenced. The link and parent (and grand parants ...) are deleted if they don't have any children and ref = -1 and they are retired.
- When the ItemNet is sending the linkDestroyed signal, it will be deleted from sync map and all queues. Finally a destroy command is sent with its netId, to spread the destruction to server and other clients.

# Chapter 5

# Example Collection

Here are some examples showing the use of the ArnLib described in this documentation.

- Chat Demo

## 5.1 Chat Demo

Demonstration with a simple chat program. It consists of a server and a client part. After starting the server, any number of clients can be started.

This demo is focused on the Service API (RPC) functionalty of ArnLib. Slots are remotely called from clients to server and the other way back. All is done with standard function calls without any visual serializing.

**Chat Server** ChatSapi.hpp, ServerMain.hpp, ServerMain.cpp, main.cpp

**Chat Client** MainWindow.hpp, MainWindow.cpp, main.cpp

### 5.1.1 Chat Server

#### 5.1.1.1 ChatSapi.hpp

```
#ifndef CHATSAPI_HPP
#define CHATSAPI_HPP

#include <ArnLib/ArnSapi.hpp>


class ChatSapi : public ArnSapi
{
    Q_OBJECT
public:
    explicit ChatSapi( QObject* parent = 0) : ArnSapi( parent)  {}

signals:
MQ_PUBLIC_ACCESS
    void  pv_list();
    void  pv_newMsg( QString name, QString msg);
    void  pv_infoQ();

    void  rq_updateMsg( int seq, QString name, QString msg);
    void  rq_info( QString name, QString ver);
};

#endif // CHATSAPI_HPP
```

#### 5.1.1.2 ServerMain.hpp

```
#ifndef SERVERMAIN_HPP
#define SERVERMAIN_HPP
```

```
#include "ChatSapi.hpp"
#include <ArnLib/ArnItem.hpp>
#include <ArnLib/ArnServer.hpp>
#include <QTimer>
#include <QStringList>
#include <QObject>


class ServerMain : public QObject
{
    Q_OBJECT
public:
    explicit ServerMain( QObject* parent = 0);

signals:

public slots:

private slots:
    void  doNewSession( QString path);
    void  doTimeUpdate();

    // Chat Provider routines
    void  chatList();
    void  chatNewMsg( QString name, QString msg);
    void  chatInfoQ();

private:
    QStringList  _chatNameList;
    QStringList  _chatMsgList;
    QTimer  _timer;

    ArnItem  _arnTime;
    ArnServer*  _server;
    ChatSapi*  _commonSapi;
};

#endif // SERVERMAIN_HPP
```

#### 5.1.1.3 ServerMain.cpp

```
#include "ServerMain.hpp"
#include <ArnLib/ArnItem.hpp>
#include <QTime>
#include <QCoreApplication>
#include <QDebug>


ServerMain::ServerMain( QObject* parent) :
    QObject( parent)
{
    _timer.start(1000);
    connect( &_timer, SIGNAL(timeout()), this, SLOT(doTimeUpdate()));

    _server = new ArnServer( ArnServer::Type::NetSync
      , this);
    _server->start();

    _arnTime.open("//Chat/Time/value");


    _commonSapi = new ChatSapi( this);
    _commonSapi->open("//Chat/Pipes/pipeCommon!", ArnSapi::Mode::Provider
      );
    _commonSapi->batchConnect( QRegExp("^pv_(.+)"), this, "chat\\1");

    ArnItem*  arnPipes = new ArnItem("//Chat/Pipes/", this);
    connect( arnPipes, SIGNAL(arnItemCreated(QString)), this, SLOT(doNewSession
      (QString)));
}


void  ServerMain::doNewSession( QString path)
{
    if (!ArnM::isProviderPath( path))  return;  // Only
       provider pipe is used

    ChatSapi*  soleSapi = new ChatSapi( this);
    soleSapi->open( path, ArnSapi::Mode::Provider);
    soleSapi->batchConnect( QRegExp("^pv_(.+)"), this, "chat\\1");
    connect( soleSapi, SIGNAL(pipeClosed()), soleSapi, SLOT(deleteLater()));
}
```

```
void  ServerMain::doTimeUpdate()
{
    _arnTime = QTime::currentTime().toString();
}


void  ServerMain::chatList()
{
    ChatSapi*  sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    for (int i = 0; i < _chatNameList.size(); ++i) {
        sapi->rq_updateMsg( i, _chatNameList.at(i), _chatMsgList.at(i));
    }
}


void  ServerMain::chatNewMsg( QString name, QString msg)
{
    _chatNameList += name;
    _chatMsgList  += msg;
    int  seq = _chatNameList.size() - 1;

    _commonSapi->rq_updateMsg( seq, name, msg);
}


void  ServerMain::chatInfoQ()
{
    ChatSapi*  sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    sapi->rq_info("Arn Chat Demo", "1.0");
}
```

#### 5.1.1.4 main.cpp

```
#include "ServerMain.hpp"
#include <QApplication>
#include <QDebug>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv, false);

    qDebug() << "Startar Arn Chat Server ...";
    new ServerMain;

    return a.exec();
}
```

### 5.1.2 Chat Client

#### 5.1.2.1 MainWindow.hpp

```
#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

#include "../ArnDemoChatServer/ChatSapi.hpp"
#include <ArnLib/ArnClient.hpp>
#include <ArnLib/ArnItem.hpp>
#include <QMainWindow>
#include <QVector>

namespace Ui {
class MainWindow;
}


class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void  doSendLine();
    void  doTimeUpdate( QString timeStr);
```

```
    // Chat Requester routines
    void  chatUpdateMsg( int seq, QString name, QString msg);
    void  chatInfo( QString name, QString ver);

private:
    Ui::MainWindow *_ui;
    QVector<QString>  _chatNameList;
    QVector<QString>  _chatMsgList;

    ArnClient  _arnClient;
    ChatSapi  _commonSapi;
    ChatSapi  _soleSapi;
    ArnItem  _arnTime;
};

#endif // MAINWINDOW_HPP
```

#### 5.1.2.2 MainWindow.cpp

```
#include "MainWindow.hpp"
#include "tmp/ui_MainWindow.h"


MainWindow::MainWindow( QWidget* parent) :
    QMainWindow( parent),
    _ui( new Ui::MainWindow)
{
    _ui->setupUi( this);
    _ui->userEdit->setFocus();
    connect( _ui->lineEdit, SIGNAL(returnPressed()), this, SLOT(doSendLine()));

    _arnClient.connectToArn("localhost");
    _arnClient.setMountPoint("//");

    _arnTime.open("//Chat/Time/value");
    connect( &_arnTime, SIGNAL(changed(QString)), this, SLOT(doTimeUpdate(
      QString)));

    _commonSapi.open("//Chat/Pipes/pipeCommon");
    _commonSapi.batchConnect( QRegExp("^rq_(.+)"), this, "chat\\1");

    _soleSapi.open("//Chat/Pipes/pipe", ArnSapi::Mode::UuidAutoDestroy
      );
    _soleSapi.batchConnect( QRegExp("^rq_(.+)"), this, "chat\\1");

    _soleSapi.pv_infoQ();
    _soleSapi.pv_list();
}


MainWindow::~MainWindow()
{
    delete _ui;
}


void  MainWindow::doTimeUpdate( QString timeStr)
{
    _ui->timeEdit->setTime( QTime::fromString( timeStr));
}


void  MainWindow::doSendLine()
{
    QString  myName = _ui->userEdit->text();
    QString  line   = _ui->lineEdit->text();
    _ui->lineEdit->clear();

    _soleSapi.pv_newMsg( myName, line);
}


void  MainWindow::chatUpdateMsg( int seq, QString name, QString msg)
{
    if (seq >= _chatNameList.size()) {
        _chatNameList.resize( seq + 1);
        _chatMsgList.resize(  seq + 1);
    }
    _chatNameList[ seq] = name;
    _chatMsgList[  seq] = msg;

    QString  text;
    for (int i = 0; i < _chatNameList.size(); ++i) {
        text += _chatNameList.at(i) + ":  " + _chatMsgList.at(i) + "\n";
```

```
    }
    _ui->textEdit->setText( text);
}


void  MainWindow::chatInfo( QString name, QString ver)
{
    _ui->appNameLabel->setText( name);
    _ui->verLabel->setText( ver);
}
```

#### 5.1.2.3 main.cpp

```
#include "MainWindow.hpp"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

### 5.1.3 Pictures

# Chapter 6

# Deprecated List

**Member ArnM::getInstance ()**

# Chapter 7

# Class Index

## 7.1 Class Hierarchy

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

# Chapter 8

# Class Index

## 8.1 Class List

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

# Chapter 9

# File Index

## 9.1 File List

Here is a list of all files with brief descriptions:

# Chapter 10

# Class Documentation

## 10.1 ArnClient Class Reference

Class for connecting to an *Arn Server*.

`#include <ArnClient.hpp>`

### Signals

- void tcpError (QString errorText, QAbstractSocket::SocketError socketError)
- void tcpConnected ()

  *Signal emitted when the tcp connection is successfull.*
- void tcpDisConnected ()

  *Signal emitted when the tcp connection is broken (has been successfull).*

### Public Member Functions

- ArnClient (QObject *parent=0)
- void connectToArn (const QString &arnHost, quint16 port=0)

  *Connect to an Arn Server*
- bool setMountPoint (const QString &path)

  *Set the sharing tree path.*
- void setAutoConnect (bool isAuto, int retryTime=2)

  *Set automatic reconnect.*

### 10.1.1 Detailed Description

Class for connecting to an *Arn Server*.

About Sharing Arn Data Objects

**Example usage**

```
// In class declare
ArnClient  _arnClient;

// In class code
_arnClient.connectToArn("localhost");
_arnClient.setMountPoint("//");
_arnClient.setAutoConnect( true);
```

**Examples:**

MainWindow.hpp.

Definition at line 63 of file ArnClient.hpp.

### 10.1.2 Constructor & Destructor Documentation

#### 10.1.2.1 ArnClient::ArnClient ( QObject * *parent* = 0 ) [explicit]

Definition at line 41 of file ArnClient.cpp.

### 10.1.3 Member Function Documentation

#### 10.1.3.1 void ArnClient::connectToArn ( const QString & *arnHost*, quint16 *port* = 0 )

Connect to an *Arn Server*

**Parameters**

| | | |
|---|---|---|
| in | *arnHost* | is host name or ip address, e.g. "192.168.1.1". |
| in | *port* | is the port number (default 2022). |

Definition at line 68 of file ArnClient.cpp.

#### 10.1.3.2 void ArnClient::setAutoConnect ( bool *isAuto*, int *retryTime* = 2 )

Set automatic reconnect.

**Parameters**

| | | |
|---|---|---|
| in | *isAuto* | true if using auto reconnect |
| in | *retryTime* | is the time between reconnection attempts in seconds |

Definition at line 91 of file ArnClient.cpp.

#### 10.1.3.3 bool ArnClient::setMountPoint ( const QString & *path* )

Set the sharing tree path.

Mountpoint is an association to the similarity of mounting a "remote filesystem". In Arn the remote "file system" is at the same sub path as the mountpoint, e.g. a client having mountpoint "/a/b/" and opening an *Arn Data Object* at "/a/b/c" will have the object *c* shared with the server at its path "/a/b/c".

**Parameters**

| | | |
|---|---|---|
| in | *path* | is the sharing tree. |

**Return values**

| | |
|---|---|
| *false* | if error. |

**See also**

Sharing Arn Data Objects

Definition at line 77 of file ArnClient.cpp.

#### 10.1.3.4 void ArnClient::tcpConnected ( ) [signal]

Signal emitted when the tcp connection is successfull.

#### 10.1.3.5 void ArnClient::tcpDisConnected ( ) [signal]

Signal emitted when the tcp connection is broken (has been successfull).

#### 10.1.3.6 void ArnClient::tcpError ( QString *errorText,* QAbstractSocket::SocketError *socketError* ) [signal]

Signal emitted when a connection tcp error occur.

**Parameters**

| | | |
|---|---|---|
| in | *errorText* | is the human readable description of the error. |
| in | *socketError* | is the error from tcp socket, see Qt doc. |

The documentation for this class was generated from the following files:

- src/ArnClient.hpp (1.0.0)
- src/ArnClient.cpp (1.0.0)

## 10.2 ArnDepend Class Reference

Class for setting up dependencis to needed services.

`#include <ArnDepend.hpp>`

### Public Types

- typedef ArnDependSlot DepSlot

### Signals

- void completed ()

  *Signal emitted when all dependent services are available.*

### Public Member Functions

- ArnDepend (QObject ∗parent=0)
- ∼ArnDepend ()
- void add (QString serviceName, int stateId=-1)

  *Add a dependency for a service*
- void add (QString serviceName, QString stateName)

  *Add a dependency for a service*
- void setMonitorName (QString name)

  *Set an optional monitor name for debugging.*
- void startMonitor ()

  *Starting the dependency monitor.*

### 10.2.1 Detailed Description

Class for setting up dependencis to needed services.

The services can be both system types available by internal Arn, and custom application types. The system types have a service name starting with "$".

This is typically used when an application needs a service to continue. When using persistent values, a client will need to know when they have been synced from the server. Then it's convenient to setup a dependency for the system service "$Persist".

When all dependent services are available, the completed() signal is emitted.

**Example usage**

```
// In class declare
ArnDepend*  _arnDepend;

// In class code
_arnDepend = new ArnDepend( this);
_arnDepend->setMonitorName("MyApp_Monitor");  // Optional for
   debug
_arnDepend->add("$Persist");
_arnDepend->add("MyService");
_arnDepend->startMonitor();
connect( _arnDepend, SIGNAL(completed()), this, SLOT(arnDependOk()
  ));
```

Definition at line 129 of file ArnDepend.hpp.

### 10.2.2 Member Typedef Documentation

#### 10.2.2.1 typedef ArnDependSlot ArnDepend::DepSlot

Definition at line 133 of file ArnDepend.hpp.

### 10.2.3 Constructor & Destructor Documentation

#### 10.2.3.1 ArnDepend::ArnDepend ( QObject * *parent* = 0 ) [explicit]

Definition at line 126 of file ArnDepend.cpp.

#### 10.2.3.2 ArnDepend::~ArnDepend ( )

Definition at line 138 of file ArnDepend.cpp.

### 10.2.4 Member Function Documentation

#### 10.2.4.1 void ArnDepend::add ( QString *serviceName*, int *stateId* = -1 )

Add a dependency for a *service*

**Parameters**

| | | |
|---|---|---|
| in | *serviceName* | is the name of the needed *service*. |
| in | *stateId* | is the needed *state* id number. -1 is don't care. |

Definition at line 172 of file ArnDepend.cpp.

#### 10.2.4.2 void ArnDepend::add ( QString *serviceName*, QString *stateName* )

Add a dependency for a *service*

**Parameters**

| | | |
|---|---|---|
| in | *serviceName* | is the name of the needed *service*. |
| in | *stateName* | is the needed *state* name. |

Definition at line 164 of file ArnDepend.cpp.

#### 10.2.4.3 void ArnDepend::completed ( ) [signal]

Signal emitted when all dependent services are available.

#### 10.2.4.4 void ArnDepend::setMonitorName ( QString *name* )

Set an optional monitor name for debugging.

**Parameters**

| | | |
|---|---|---|
| in | *name* | is the monitor name. |

Definition at line 180 of file ArnDepend.cpp.

#### 10.2.4.5 void ArnDepend::startMonitor ( )

Starting the dependency monitor.

Definition at line 186 of file ArnDepend.cpp.

The documentation for this class was generated from the following files:

- src/ArnDepend.hpp (1.0.0)
- src/ArnDepend.cpp (1.0.0)

## 10.3 ArnDependOffer Class Reference

Class for advertising that a *service* is available.

`#include <ArnDepend.hpp>`

### Public Member Functions

- ArnDependOffer (QObject *parent=0)
- void advertise (QString serviceName)

  *Advertise an available service*
- void setStateName (const QString &name)

  *Set the state of the service by a logic name.*
- QString stateName () const
- void setStateId (int id)

  *Set the state of the service by an id number.*
- int stateId () const

### 10.3.1 Detailed Description

Class for advertising that a *service* is available.

Additionally it's possible to indicate the *state* of the *service*. The *state* can either be indicated by a logic name or by an id number whichever is prefered.

**Example usage**

```
// In class declare
ArnDependOffer* _depOffer;

// In class code
_depOffer = new ArnDependOffer( this);
_depOffer->advertise("MyService");  // Service now available
```

Definition at line 61 of file ArnDepend.hpp.

### 10.3.2 Constructor & Destructor Documentation

#### 10.3.2.1 ArnDependOffer::ArnDependOffer ( QObject * *parent* = 0 ) [explicit]

Definition at line 45 of file ArnDepend.cpp.

### 10.3.3 Member Function Documentation

#### 10.3.3.1 void ArnDependOffer::advertise ( QString *serviceName* )

Advertise an available *service*

**Parameters**

| | | |
|---|---|---|
| in | *serviceName* | is the name of the *service*. |

Definition at line 51 of file ArnDepend.cpp.

#### 10.3.3.2 void ArnDependOffer::setStateId ( int *id* )

Set the *state* of the *service* by an id number.

The *state* starts of by 0 as default.

**Parameters**

| | | |
|---|---|---|
| in | *id* | is the *state* id number. |

Definition at line 83 of file ArnDepend.cpp.

#### 10.3.3.3 void ArnDependOffer::setStateName ( const QString & *name* )

Set the *state* of the *service* by a logic name.

The *state* starts of by "Start" as default.

**Parameters**

| | | |
|---|---|---|
| in | *name* | is the *state* name. |

Definition at line 71 of file ArnDepend.cpp.

#### 10.3.3.4 int ArnDependOffer::stateId ( ) const

**Returns**

The *state* id number.

**See also**

setStateId()

Definition at line 89 of file ArnDepend.cpp.

#### 10.3.3.5 QString ArnDependOffer::stateName ( ) const

**Returns**

The logic *state* name, e.g. the default "Start"

**See also**

setStateName()

Definition at line 77 of file ArnDepend.cpp.

The documentation for this class was generated from the following files:

- src/ArnDepend.hpp (1.0.0)
- src/ArnDepend.cpp (1.0.0)

## 10.4 ArnError Struct Reference

`#include <ArnError.hpp>`

### Classes

- struct StdCode

### Public Types

- enum E {
  Ok = 0, Info = StdCode::Info, Warning = StdCode::Warning, Undef = StdCode::Err_Undef,
  CreateError = StdCode::Err_Custom, NotFound, NotOpen, AlreadyExist,
  AlreadyOpen, Retired, NotMainThread, FolderNotOpen,
  ItemNotOpen, ItemNotSet, ConnectionError, RecUnknown,
  ScriptError, RpcInvokeError, RpcReceiveError, Err_N }

### 10.4.1 Detailed Description

Definition at line 39 of file ArnError.hpp.

### 10.4.2 Member Enumeration Documentation

#### 10.4.2.1 enum ArnError::E

**Enumerator:**

***Ok***

***Info***

***Warning***

***Undef***

***CreateError***

***NotFound***

***NotOpen***

***AlreadyExist***

***AlreadyOpen***

***Retired***

***NotMainThread***

***FolderNotOpen***

***ItemNotOpen***

***ItemNotSet***

***ConnectionError***

***RecUnknown***

***ScriptError***

***RpcInvokeError***

***RpcReceiveError***

***Err_N***

Definition at line 52 of file ArnError.hpp.

The documentation for this struct was generated from the following file:

- src/ArnError.hpp (1.0.0)

## 10.5 ArnItem Class Reference

Handle for an *Arn Data Object*.

`#include <ArnItem.hpp>`

### Classes

- struct Mode

  *General global mode of an Arn Data Object*
- struct SyncMode

  *The client session sync mode of an Arn Data Object*

## Public Slots

- void setValue (int value, int ignoreSame=-1)

  *Assign an integer to an Arn Data Object*
- void setValue (double value, int ignoreSame=-1)

  *Assign a double to an Arn Data Object*
- void setValue (bool value, int ignoreSame=-1)

  *Assign a bool to an Arn Data Object*
- void setValue (const QString &value, int ignoreSame=-1)

  *Assign a QString to an Arn Data Object*
- void setValue (const QByteArray &value, int ignoreSame=-1)

  *Assign a QByteArray to an Arn Data Object*
- void setValue (const QVariant &value, int ignoreSame=-1)

  *Assign a QVariant to an Arn Data Object*
- void setValue (const char *value, int ignoreSame=-1)

  *Assign a char* to an Arn Data Object*
- void toggleBool ()

  *Toggle the bool at the Arn Data Object*

## Signals

- void changed ()

  *Signals emitted when Arn Data Object is changed.*
- void changed (int value)
- void changed (double value)
- void changed (bool value)
- void changed (QString value)
- void changed (QByteArray value)
- void changed (QVariant value)
- void arnItemCreated (QString path)

  *Signal emitted when an Arn Data Object is created in the tree below.*
- void arnModeChanged (QString path, uint linkId, ArnItem::Mode mode)

  *Signal emitted when an Arn Data Object in the tree below has a general mode change.*
- void arnLinkDestroyed ()

  *Signal emitted when the Arn Data Object is destroyed.*

## Public Member Functions

- ArnItem (QObject *parent=0)

  *Standard constructor of a closed handle.*
- ArnItem (const QString &path, QObject *parent=0)

  *Construction of a handle to a path.*
- ArnItem (const ArnItem &folder_template, const QString &itemName_path, QObject *parent=0)

  *Construction of a handle to a path with a template for modes*
- virtual ~ArnItem ()
- bool open (const QString &path)

  *Open a handle to an Arn Data Object*
- bool openUuidPipe (const QString &path)

  *Open a handle to an Arn Pipe Object with a unique uuid name.*
- bool openFolder (const QString &path)

  *Open a handle to an Arn folder.*

- void close ()

  *Close the handle.*
- void destroyLink ()

  *Destroy the Arn Data Object*
- bool isOpen () const

  *State of the handle.*
- bool isFolder () const
- bool isBiDir () const
- ArnLink::Type type () const

  *The type stored in the Arn Data Object*
- QString path (ArnLink::NameF nameF=ArnLink::NameF::EmptyOk) const

  *Path of the Arn Data Object*
- QString name (ArnLink::NameF nameF) const

  *Name of the Arn Data Object*
- bool isOnlyEcho () const
- void setBlockEcho (bool blockEcho)
- void setIgnoreSameValue (bool isIgnore=true)

  *Set skipping of equal value.*
- bool isIgnoreSameValue ()
- void setReference (void *reference)

  *Set an associated external reference.*
- void * reference () const

  *Get the stored external reference.*
- uint itemId () const

  *Get the id for this ArnItem.*
- uint linkId () const

  *Get the id for this Arn Data Object*
- void addMode (Mode mode)

  *Add general mode settings for this Arn Data Object*
- Mode getMode () const

  *Use with care, link must be "referenced" before use, otherwise it might have been deleted.*
- SyncMode syncMode () const
- ArnItem & setTemplate (bool isTemplate=true)

  *Mark this ArnItem as a template.*
- bool isTemplate () const
- ArnItem & setBiDirMode ()

  *Set general mode as Bidirectional for this Arn Data Object*
- bool isBiDirMode () const
- ArnItem & setPipeMode ()

  *Set general mode as Pipe for this Arn Data Object*
- bool isPipeMode () const
- ArnItem & setSaveMode ()

  *Set general mode as Save for this Arn Data Object*
- bool isSaveMode () const
- ArnItem & setMaster ()

  *Set client session sync mode as Master for this ArnItem.*
- bool isMaster () const
- ArnItem & setAutoDestroy ()

  *Set client session sync mode as AutoDestroy for this ArnItem.*
- bool isAutoDestroy () const
- void setDelay (int delay)

*Set delay of data changed signal.*

- void arnImport (const QByteArray &data, int ignoreSame=-1)

  *Import data to an Arn Data Object*
- QByteArray arnExport () const
- int toInt () const
- double toDouble () const
- bool toBool () const
- QString toString () const
- QByteArray toByteArray () const
- QVariant toVariant () const
- ArnItem & operator= (const ArnItem &other)
- ArnItem & operator= (int other)
- ArnItem & operator= (double other)
- ArnItem & operator= (const QString &other)
- ArnItem & operator= (const QByteArray &other)
- ArnItem & operator= (const QVariant &other)
- ArnItem & operator= (const char *other)

## Friends

- class ArnClient
- class ArnSync

### 10.5.1 Detailed Description

Handle for an *Arn Data Object*.

About Arn Data Object

When opening an ArnItem to an *Arn Data object*, the ArnItem act as a handle (pointer) to the object. There can be any amount of ArnItem:s opened (pointing) to the same *Arn Data object*. Deleting the ArnItem won't effect the *Arn Data object*.

This class is not thread-safe, but the *Arn Data object* is, so each thread should have it's own handles i.e ArnItem instances.

**Example usage**

```
// In class declare
ArnItem  _arnTime;

// In class code
_arnTime.open("//Chat/Time/value");
connect( &_arnTime, SIGNAL(changed(QString)), this, SLOT(
  doTimeUpdate(QString)));
_arnTime = "Undefined ...";
```

**Examples:**

MainWindow.hpp, ServerMain.cpp, and ServerMain.hpp.

Definition at line 71 of file ArnItem.hpp.

### 10.5.2 Constructor & Destructor Documentation

#### 10.5.2.1 ArnItem::ArnItem ( QObject * *parent* = 0 )

Standard constructor of a closed handle.

Definition at line 71 of file ArnItem.cpp.

#### 10.5.2.2 ArnItem::ArnItem ( const QString & *path*, QObject * *parent* = 0 )

Construction of a handle to a path.

**Parameters**

| | | |
|---|---|---|
| in | *path* | The *Arn Data Object* path e.g. "//Measure/Water/Level/value" |

**See also**

open()

Definition at line 78 of file ArnItem.cpp.

#### 10.5.2.3 ArnItem::ArnItem ( const ArnItem & *folder_template*, const QString & *itemName_path*, QObject * *parent* = 0 )

Construction of a handle to a path with a template for *modes*

**Parameters**

| | | |
|---|---|---|
| in | *path* | The *Arn Data Object* path e.g. "//Measure/Water/Level/value" |
| in | *folder_template* | The template for setting *modes* |

Double usage 2 modes: template, folder

Definition at line 86 of file ArnItem.cpp.

#### 10.5.2.4 ArnItem::~ArnItem ( ) [virtual]

Definition at line 980 of file ArnItem.cpp.

### 10.5.3 Member Function Documentation

#### 10.5.3.1 void ArnItem::addMode ( Mode *mode* )

Add *general mode* settings for this *Arn Data Object*

If this ArnItem is in closed state, the added modes will be stored and the real mode change is done when this ArnItem is opened to an *Arn Data Object*. This implies that ArnItems can benefit from setting *modes* before opening.

**Parameters**

| | | |
|---|---|---|
| in | *mode* | The *modes* to be added. |

**See also**

getMode()
Modes

Definition at line 388 of file ArnItem.cpp.

#### 10.5.3.2 QByteArray ArnItem::arnExport ( ) const

**Returns**

A data blob representing the *Arn Data Object*

**See also**

arnImport()

Definition at line 489 of file ArnItem.cpp.

#### 10.5.3.3 void ArnItem::arnImport ( const QByteArray & *data*, int *ignoreSame* = -1 )

Import data to an *Arn Data Object*

Data blob from a previos `arnExport()` can be imported. This is essentially assigning the *Arn Data Object* with same as exported.

**Parameters**

| | | |
|---|---|---|
| in | *data* | is the data blob |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

arnExport()
setIgnoreSameValue()

Definition at line 462 of file ArnItem.cpp.

#### 10.5.3.4 void ArnItem::arnItemCreated ( QString *path* ) [signal]

Signal emitted when an *Arn Data Object* is created in the tree below.

The ArnItem is a folder. Created objects in this folder or its children will give this signal. Only created non folder objects will give this signal.

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the created *Arn Data Object* |

#### 10.5.3.5 void ArnItem::arnLinkDestroyed ( ) [signal]

Signal emitted when the *Arn Data Object* is destroyed.

When the link (*Arn Data Object*) is destroyed, this ArnItem is closed and will give this signal. It's ok to assign values etc to a closed ArnItem, it's thrown away like a null device.

**See also**

destroyLink()

#### 10.5.3.6 void ArnItem::arnModeChanged ( QString *path*, uint *linkId*, ArnItem::Mode *mode* ) [signal]

Signal emitted when an *Arn Data Object* in the tree below has a *general mode* change.

The ArnItem is a folder. Objects changing *general mode* in this folder or its children will give this signal.

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the *general mode* changing *Arn Data Object* |
| in | *linkId* | for the *general mode* changing *Arn Data Object* |
| in | *mode* | is the new *general mode* |

**See also**

linkId()
Modes

#### 10.5.3.7 void ArnItem::changed ( ) [signal]

Signals emitted when *Arn Data Object* is changed.

Only the connected (used) signals are emitted for efficiency. When using pipes with queued connection to a slot, it's strongly advised to use the signal that carries the updated data. Otherwise some stream data can be lost and other will be doubled, because reading is done late in the slot.

changed(...) is using connectNotify & disconnectNotify. Must be updated if new types are added

**See also**

setIgnoreSameValue()

#### 10.5.3.8 void ArnItem::changed ( int *value* ) [signal]

**See also**

changed()

#### 10.5.3.9 void ArnItem::changed ( double *value* ) [signal]

**See also**

changed()

#### 10.5.3.10 void ArnItem::changed ( bool *value* ) [signal]

**See also**

changed()

#### 10.5.3.11 void ArnItem::changed ( QString *value* ) [signal]

**See also**

changed()

#### 10.5.3.12 void ArnItem::changed ( QByteArray *value* ) [signal]

**See also**

changed()

#### 10.5.3.13 void ArnItem::changed ( QVariant *value* ) [signal]

**See also**

changed()

#### 10.5.3.14 void ArnItem::close ( )

Close the handle.

Definition at line 181 of file ArnItem.cpp.

#### 10.5.3.15 void ArnItem::destroyLink ( )

Destroy the *Arn Data Object*

The link (*Arn Data Object*) will be removed locally, from server and all connected clients.

Definition at line 192 of file ArnItem.cpp.

#### 10.5.3.16 ArnItem::Mode ArnItem::getMode ( ) const

Use with care, link must be "referenced" before use, otherwise it might have been deleted.

**Returns**

The *general mode* of the *Arn Data Object*

**See also**

addMode()
Modes

Definition at line 404 of file ArnItem.cpp.

#### 10.5.3.17 bool ArnItem::isAutoDestroy ( ) const

**Return values**

| | |
|---|---|
| *true* | if *AutoDestroy mode* |

**See also**

setAutoDestroy()

Definition at line 382 of file ArnItem.cpp.

#### 10.5.3.18 bool ArnItem::isBiDir ( ) const

**Return values**

| | |
|---|---|
| *true* | if this ArnItem is bi-directional |

**See also**

setBiDirMode()
Modes

Definition at line 218 of file ArnItem.cpp.

#### 10.5.3.19 bool ArnItem::isBiDirMode ( ) const

**Return values**

| | |
|---|---|
| *true* | if Bidirectional |

**See also**

setBiDirMode()
Modes
Bidirectional Arn Data Objects

Definition at line 303 of file ArnItem.cpp.

#### 10.5.3.20 bool ArnItem::isFolder ( ) const

**Return values**

| | |
|---|---|
| *true* | if this ArnItem is a folder |

Definition at line 210 of file ArnItem.cpp.

#### 10.5.3.21 bool ArnItem::isIgnoreSameValue ( )

**Return values**

| | |
|---|---|
| *true* | if skipping equal values |

**See also**

setIgnoreSameValue()

Definition at line 430 of file ArnItem.cpp.

#### 10.5.3.22 bool ArnItem::isMaster ( ) const

**Return values**

| | |
|---|---|
| *true* | if *Master mode* |

**See also**

setMaster()
Modes

Definition at line 365 of file ArnItem.cpp.

#### 10.5.3.23 bool ArnItem::isOnlyEcho ( ) const [inline]

Definition at line 181 of file ArnItem.hpp.

#### 10.5.3.24 bool ArnItem::isOpen ( ) const

State of the handle.

**Return values**

| | |
|---|---|
| *true* | if this ArnItem is open |

Definition at line 198 of file ArnItem.cpp.

#### 10.5.3.25 bool ArnItem::isPipeMode ( ) const

**Return values**

| | |
|---|---|
| *true* | if *Pipe mode* |

**See also**

setPipeMode()
Modes
Pipes

Definition at line 328 of file ArnItem.cpp.

#### 10.5.3.26 bool ArnItem::isSaveMode ( ) const

**Return values**

| | |
|---|---|
| *true* | if *Save mode* |

**See also**

setSaveMode()
Modes
Persistent Arn Data Objects

Definition at line 346 of file ArnItem.cpp.

#### 10.5.3.27 bool ArnItem::isTemplate ( ) const

**Return values**

| | |
|---|---|
| *true* | if this is a template |

**See also**

setTemplate()

Definition at line 282 of file ArnItem.cpp.

#### 10.5.3.28 uint ArnItem::itemId ( ) const [inline]

Get the *id* for this ArnItem.

The ArnItem *id* is unique within its running program. Even if 2 ArnItems are pointing to the same *Arn Data Object*, they have different *item id*.

**Returns**

*id* for this ArnItem

**See also**

linkId()

Definition at line 216 of file ArnItem.hpp.

#### 10.5.3.29 uint ArnItem::linkId ( ) const

Get the *id* for this *Arn Data Object*

The link (*Arn Data Object*) *id* is unique within its running program. If 2 ArnItems are pointing to the same *Arn Data Object*, they have same *link id*.

**Returns**

Id for the *Arn Data Object*, 0 if closed

**See also**

itemId()

Definition at line 234 of file ArnItem.cpp.

#### 10.5.3.30 QString ArnItem::name ( ArnLink::NameF *nameF* ) const

Name of the *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *nameF* | The format of the returned name |

**Returns**

The object name

Definition at line 444 of file ArnItem.cpp.

#### 10.5.3.31 bool ArnItem::open ( const QString & *path* )

Open a handle to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *path* | The *Arn Data Object* path e.g. "//Measure/Water/Level/value" |

**Return values**

| | |
|---|---|
| *false* | if error |

Definition at line 136 of file ArnItem.cpp.

#### 10.5.3.32 bool ArnItem::openFolder ( const QString & *path* )

Open a handle to an Arn folder.

**Parameters**

| | | |
|---|---|---|
| in | *path* | The Arn folder path e.g. "//Measure/Water" (the / is appended) |

**Return values**

| | |
|---|---|
| *false* | if error |

Definition at line 153 of file ArnItem.cpp.

#### 10.5.3.33 bool ArnItem::openUuidPipe ( const QString & *path* )

Open a handle to an Arn Pipe Object with a unique uuid name.

**Parameters**

| | | |
|---|---|---|
| in | *path* | The prefix for Arn uuid pipe path e.g. "//Pipes/pipe" |

**Return values**

| | |
|---|---|
| *false* | if error |

Definition at line 143 of file ArnItem.cpp.

#### 10.5.3.34 ArnItem & ArnItem::operator= ( const ArnItem & *other* )

Definition at line 564 of file ArnItem.cpp.

#### 10.5.3.35 ArnItem & ArnItem::operator= ( int *other* )

Definition at line 608 of file ArnItem.cpp.

#### 10.5.3.36 ArnItem & ArnItem::operator= ( double *other* )

Definition at line 615 of file ArnItem.cpp.

#### 10.5.3.37 ArnItem & ArnItem::operator= ( const QString & *other* )

Definition at line 622 of file ArnItem.cpp.

#### 10.5.3.38 ArnItem & ArnItem::operator= ( const QByteArray & *other* )

Definition at line 629 of file ArnItem.cpp.

#### 10.5.3.39 ArnItem & ArnItem::operator= ( const QVariant & *other* )

Definition at line 643 of file ArnItem.cpp.

#### 10.5.3.40 ArnItem & ArnItem::operator= ( const char * *other* )

Definition at line 636 of file ArnItem.cpp.

#### 10.5.3.41 QString ArnItem::path ( ArnLink::NameF *nameF* = ArnLink::NameF::EmptyOk ) const

Path of the *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *nameF* | The format of the returned path |

**Returns**

The object path

Definition at line 436 of file ArnItem.cpp.

#### 10.5.3.42 void* ArnItem::reference ( ) const [inline]

Get the stored external reference.

**Returns**

The associated external reference

**See also**

setReference()

Definition at line 208 of file ArnItem.hpp.

#### 10.5.3.43 ArnItem & ArnItem::setAutoDestroy ( )

Set client session *sync mode* as *AutoDestroy* for this ArnItem.

This ArnItem at client side is setup for auto destruction.

**Precondition**

This must be set before open().

Definition at line 371 of file ArnItem.cpp.

#### 10.5.3.44 ArnItem & ArnItem::setBiDirMode ( )

Set *general mode* as Bidirectional for this *Arn Data Object*

A two way object, typically for validation or pipe

**See also**

Modes
Bidirectional Arn Data Objects

Bidirectional-mode is the pair of value & provider

Definition at line 288 of file ArnItem.cpp.

#### 10.5.3.45 void ArnItem::setBlockEcho ( bool *blockEcho* ) [inline]

Definition at line 182 of file ArnItem.hpp.

#### 10.5.3.46 void ArnItem::setDelay ( int *delay* )

Set *delay* of data changed signal.

Normally any change of the *Arn Data Object* is immediately signalled. By setting this *delay*, intensive updates gives predictive and fewer signals. Signalling will not be faster than *delay* as period time. The latency from a change to a signal will not be more than *delay*.

**Parameters**

| | | |
|---|---|---|
| in | *delay* | in ms. |

Definition at line 452 of file ArnItem.cpp.

#### 10.5.3.47 void ArnItem::setIgnoreSameValue ( bool *isIgnore* = `true` )

Set skipping of equal value.

**Parameters**

| | | |
|---|---|---|
| in | *isIgnore* | If true, assignment of equal value don't give a changed signal. |

Definition at line 424 of file ArnItem.cpp.

#### 10.5.3.48 ArnItem & ArnItem::setMaster ( )

Set client session *sync mode* as *Master* for this ArnItem.

This ArnItem at client side is set as default generator of data.

**Precondition**

This must be set before open().

**See also**

Modes

Definition at line 354 of file ArnItem.cpp.

#### 10.5.3.49 ArnItem & ArnItem::setPipeMode ( )

Set *general mode* as Pipe for this *Arn Data Object*

Implies *Bidir*.

**See also**

Modes
Pipes

Definition at line 311 of file ArnItem.cpp.

#### 10.5.3.50 void ArnItem::setReference ( void * *reference* ) `[inline]`

Set an associated external reference.

This is typically used when having many *ArnItems* changed signal connected to a common slot. The slot can then discover the signalling ArnItem:s associated structure for further processing.

**Parameters**

| | | |
|---|---|---|
| in | *reference* | Any external structure or id. |

**See also**

reference()

Definition at line 202 of file ArnItem.hpp.

#### 10.5.3.51 ArnItem & ArnItem::setSaveMode ( )

Set *general mode* as *Save* for this *Arn Data Object*

Data is persistent and will be saved

**Precondition**

The persistent service must be started at the server.

**See also**

Modes
Persistent Arn Data Objects

Definition at line 336 of file ArnItem.cpp.

#### 10.5.3.52 ArnItem & ArnItem::setTemplate ( bool *isTemplate* = `true` )

Mark this ArnItem as a template.

When marked as a template it can be setup with a combination of *modes* which are used for other ArnItems using this template. The effected *modes* can be both *general modes* and *sync modes*.

**Parameters**

| | | |
|---|---|---|
| in | *isTemplate* | True for template mode. |

**See also**

open()
Modes

Definition at line 275 of file ArnItem.cpp.

#### 10.5.3.53 void ArnItem::setValue ( int *value,* int *ignoreSame* = `-1` ) `[slot]`

Assign an *integer* to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 650 of file ArnItem.cpp.

#### 10.5.3.54 void ArnItem::setValue ( double *value,* int *ignoreSame* = −1 ) [slot]

Assign a *double* to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 671 of file ArnItem.cpp.

#### 10.5.3.55 void ArnItem::setValue ( bool *value,* int *ignoreSame* = −1 ) [slot]

Assign a *bool* to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 692 of file ArnItem.cpp.

#### 10.5.3.56 void ArnItem::setValue ( const QString & *value,* int *ignoreSame* = −1 ) [slot]

Assign a *QString* to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 713 of file ArnItem.cpp.

#### 10.5.3.57 void ArnItem::setValue ( const QByteArray & *value,* int *ignoreSame* = −1 ) [slot]

Assign a *QByteArray* to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 734 of file ArnItem.cpp.

#### 10.5.3.58 void ArnItem::setValue ( const QVariant & *value*, int *ignoreSame* = -1 ) [slot]

Assign a *QVariant* to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 755 of file ArnItem.cpp.

#### 10.5.3.59 void ArnItem::setValue ( const char * *value*, int *ignoreSame* = -1 ) [slot]

Assign a *char** to an *Arn Data Object*

**Parameters**

| | | |
|---|---|---|
| in | *value* | to be assigned |
| in | *ignoreSame* | -1 = don't care, otherwise overide ignoreSame setting. |

**See also**

setIgnoreSameValue()

Definition at line 776 of file ArnItem.cpp.

#### 10.5.3.60 ArnItem::SyncMode ArnItem::syncMode ( ) const

**Returns**

The client session *sync mode* of an *Arn Data Object*

**See also**

addSyncMode()
Modes

Definition at line 260 of file ArnItem.cpp.

#### 10.5.3.61 bool ArnItem::toBool ( ) const

**Returns**

Convert *Arn Data Object* to a *bool*

Definition at line 556 of file ArnItem.cpp.

#### 10.5.3.62 QByteArray ArnItem::toByteArray ( ) const

**Returns**

Convert *Arn Data Object* to a *QByteArray*

Definition at line 524 of file ArnItem.cpp.

#### 10.5.3.63 double ArnItem::toDouble ( ) const

**Returns**

Convert *Arn Data Object* to a *double*

Definition at line 548 of file ArnItem.cpp.

#### 10.5.3.64 void ArnItem::toggleBool ( ) [slot]

Toggle the *bool* at the *Arn Data Object*

The *Arn Data Object* is first converted to a *bool*, then the toggled value is assigned back to the *Arn Data Object*.

Definition at line 782 of file ArnItem.cpp.

#### 10.5.3.65 int ArnItem::toInt ( ) const

**Returns**

Convert *Arn Data Object* to a *integer*

Definition at line 540 of file ArnItem.cpp.

#### 10.5.3.66 QString ArnItem::toString ( ) const

**Returns**

Convert *Arn Data Object* to a *QString*

Definition at line 516 of file ArnItem.cpp.

#### 10.5.3.67 QVariant ArnItem::toVariant ( ) const

**Returns**

Convert *Arn Data Object* to a *QVariant*

Definition at line 532 of file ArnItem.cpp.

#### 10.5.3.68 ArnLink::Type ArnItem::type ( ) const

The type stored in the *Arn Data Object*

**Returns**

The type stored

Definition at line 226 of file ArnItem.cpp.

### 10.5.4 Friends And Related Function Documentation

#### 10.5.4.1 friend class ArnClient [friend]

Definition at line 74 of file ArnItem.hpp.

#### 10.5.4.2 friend class ArnSync [friend]

Definition at line 75 of file ArnItem.hpp.

The documentation for this class was generated from the following files:

- src/ArnItem.hpp (1.0.0)
- src/ArnItem.cpp (1.0.0)

## 10.6 ArnLink Class Reference

```
#include <ArnLink.hpp>
```

### Classes

- struct Flags
- struct NameF
- struct Type

### Friends

- class ArnM

### 10.6.1 Detailed Description

Definition at line 46 of file ArnLink.hpp.

### 10.6.2 Friends And Related Function Documentation

#### 10.6.2.1 friend class ArnM [friend]

Definition at line 50 of file ArnLink.hpp.

The documentation for this class was generated from the following files:

- src/ArnLink.hpp (1.0.0)
- src/ArnLink.cpp (1.0.0)

## 10.7 ArnM Class Reference

```
#include <Arn.hpp>
```

### Public Slots

- static void destroyLink (const QString &path)

  *Destroy the Arn Data Object at path*
- static void setupErrorlog (QObject *errLog)

### Signals

- void errorLogSig (QString errText, uint errCode, void *reference)

### Static Public Member Functions

- static ArnM & instance ()
- static ArnM & getInstance ()
- static void setConsoleError (bool isConsoleError)
- static void setDefaultIgnoreSameValue (bool isIgnore=true)

  *Set system default skipping of equal value.*
- static bool defaultIgnoreSameValue ()
- static bool isMainThread ()
- static bool isThreadedApp ()
- static bool isProviderPath (const QString &path)

  *Test if path is a provider path*
- static QString itemName (const QString &path)
- static QString childPath (const QString &parentPath, const QString &posterityPath)

  *Get substring for child from a path.*
- static QString makePath (const QString &parentPath, const QString &itemName)

  *Make a path from a parent and an item name.*
- static QString addPath (const QString &parentPath, const QString &childRelPath, ArnLink::NameF nameF=ArnLink::NameF::EmptyOk)

  *Make a path from a parent and an additional relative path.*
- static QString convertPath (const QString &path, ArnLink::NameF nameF=ArnLink::NameF::EmptyOk)

  *Convert a path to a specific format.*
- static QString twinPath (const QString &path)

  *Get the bidirectional twin to a given path*
- static int valueInt (const QString &path)

  *Get the value of Arn Data Object at path*
- static double valueDouble (const QString &path)

  *Get the value of Arn Data Object at path*
- static QString valueString (const QString &path)

  *Get the value of Arn Data Object at path*
- static QByteArray valueByteArray (const QString &path)

  *Get the value of Arn Data Object at path*
- static QVariant valueVariant (const QString &path)

  *Get the value of Arn Data Object at path*
- static QStringList items (const QString &path)

  *Get the childrens of the folder at path*
- static bool exist (const QString &path)
- static bool isFolder (const QString &path)
- static bool isLeaf (const QString &path)
- static void setValue (const QString &path, int value)

  *Assign an integer to an Arn Data Object at path*

- static void setValue (const QString &path, double value)

  *Assign a double to an Arn Data Object at path*
- static void setValue (const QString &path, const QString &value)

  *Assign a QString to an Arn Data Object at path*
- static void setValue (const QString &path, const QByteArray &value)

  *Assign a QByteArray to an Arn Data Object at path*
- static void setValue (const QString &path, const QVariant &value)

  *Assign a QVariant to an Arn Data Object at path*
- static void errorLog (QString errText, ArnError err=ArnError::Undef, void *reference=0)
- static QString errorSysName ()
- static QByteArray info ()

  *Give information about this library.*

### Friends

- class ArnItem

### 10.7.1 Detailed Description

Arn main class

About Arn Data Object

This singleton class is the main reference to the Active Registry Network.

Definition at line 101 of file Arn.hpp.

### 10.7.2 Member Function Documentation

#### 10.7.2.1 QString ArnM::addPath ( const QString & *parentPath,* const QString & *childRelPath,* ArnLink::NameF *nameF* = ArnLink::NameF::EmptyOk ) [static]

Make a path from a parent and an additional relative path.

*parentPath* don't have to end with a "/", if missing it's added.

Example: *parentPath* = "//Measure/", *childRelPath* = "depth/value" ==> return = "//Measure/depth/value"

**Parameters**

| | | |
|---|---|---|
| in | *parentPath* | |
| in | *childRelPath* | |
| in | *nameF* | is the path naming format |

**Returns**

The *path*

**See also**

convertPath()

Definition at line 290 of file Arn.cpp.

#### 10.7.2.2 QString ArnM::childPath ( const QString & *parentPath,* const QString & *posterityPath* ) [static]

Get substring for child from a path.

*parentPath* don't have to end with a "/", if missing it's added.

If *posterityPath* not starts with *parentPath*, QString() is returned. Otherwise given the *posterityPath* the child to *parentPath* is returned.

Example 1: *posterityPath* = "//Measure/depth/value", *parentPath* = "//Measure/" ==> return = "//Measure/depth/"

Example 2: *posterityPath* = "//Measure/depth/value", *parentPath* = "//Measure/depth/" ==> return = //-Measure/depth/value"

**Parameters**

| | | |
|---|---|---|
| in | *parentPath* | |
| in | *posterityPath* | |

**Returns**

The *child path*

Definition at line 267 of file Arn.cpp.

#### 10.7.2.3 QString ArnM::convertPath ( const QString & *path*, ArnLink::NameF *nameF* = ArnLink::NameF::EmptyOk ) [static]

Convert a path to a specific format.

Example: *path* = "//Measure/depth/value", nameF = Relative ==> return = "@/Measure/depth/value"

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *nameF* | is the path naming format |

**Returns**

The converted *path*

Definition at line 301 of file Arn.cpp.

#### 10.7.2.4 bool ArnM::defaultIgnoreSameValue ( ) [static]

**Return values**

| | |
|---|---|
| *true* | if default skipping equal values |

**See also**

setDefaultIgnoreSameValue()

Definition at line 914 of file Arn.cpp.

#### 10.7.2.5 void ArnM::destroyLink ( const QString & *path* ) [static], [slot]

Destroy the *Arn Data Object* at *path*

The link (*Arn Data Object*) will be removed locally, from server and all connected clients.

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

Threaded version of destroyLink

Definition at line 713 of file Arn.cpp.

#### 10.7.2.6 void ArnM::errorLog ( QString *errText,* ArnError *err* = ArnError::Undef, void * *reference* = 0 ) [static]

Definition at line 831 of file Arn.cpp.

#### 10.7.2.7 void ArnM::errorLogSig ( QString *errText,* uint *errCode,* void * *reference* ) [signal]

#### 10.7.2.8 QString ArnM::errorSysName ( ) [static]

Definition at line 791 of file Arn.cpp.

#### 10.7.2.9 bool ArnM::exist ( const QString & *path* ) [static]

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Return values**

| | |
|---|---|
| *true* | if *Arn Data Object* exist at *path* |

Definition at line 351 of file Arn.cpp.

#### 10.7.2.10 static ArnM& ArnM::getInstance ( ) [inline], [static]

**Deprecated**

Definition at line 110 of file Arn.hpp.

#### 10.7.2.11 QByteArray ArnM::info ( ) [static]

Give information about this library.

**Returns**

The info, e.g. "Name=ArnLib Ver=1.0.0 Date=12-12-30 Time=00:37"

Definition at line 797 of file Arn.cpp.

#### 10.7.2.12 ArnM & ArnM::instance ( ) [static]

Definition at line 894 of file Arn.cpp.

#### 10.7.2.13 bool ArnM::isFolder ( const QString & *path* ) [static]

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Return values**

| | |
|---|---|
| *true* | if *Arn Data Object* at *path* is a folder |

Definition at line 362 of file Arn.cpp.

#### 10.7.2.14 bool ArnM::isLeaf ( const QString & *path* ) [static]

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Return values**

| | |
|---|---|
| *true* | if *Arn Data Object* at *path* is a leaf (non folder) |

Definition at line 373 of file Arn.cpp.

#### 10.7.2.15 bool ArnM::isMainThread ( ) [static]

**Return values**

| | |
|---|---|
| *true* | if this is the main thread in the application |

Definition at line 235 of file Arn.cpp.

#### 10.7.2.16 bool ArnM::isProviderPath ( const QString & *path* ) [static]

Test if *path* is a *provider path*

About Bidirectional Arn Data Objects

**Parameters**

| | | |
|---|---|---|
| in | *path.* | |

**Return values**

| | |
|---|---|
| *true* | if *path* is a *provider path*, i.e. ends with a "!". |

**Examples:**

ServerMain.cpp.

Definition at line 345 of file Arn.cpp.

#### 10.7.2.17 bool ArnM::isThreadedApp ( ) [static]

**Return values**

| | |
|---|---|
| *true* | if this is a threaded application |

Definition at line 251 of file Arn.cpp.

#### 10.7.2.18 QString ArnM::itemName ( const QString & *path* ) [static]

**Returns**

The itemName, i.e. the last part of the path after last "/"

Definition at line 257 of file Arn.cpp.

#### 10.7.2.19 QStringList ArnM::items ( const QString & *path* ) [static]

Get the childrens of the folder at *path*

Example: return list = {"test"; "folder/"; "@/"; "value"}

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The items (children)

Definition at line 175 of file Arn.cpp.

#### 10.7.2.20 QString ArnM::makePath ( const QString & *parentPath,* const QString & *itemName* ) [static]

Make a path from a parent and an item name.

*parentPath* don't have to end with a "/", if missing it's added. Empty folder *itemName* is allowed on returned path.

Example: *parentPath* = "//Measure/depth/", *itemName* = "value" ==> return = "//Measure/depth/value"

**Parameters**

| | | |
|---|---|---|
| in | *parentPath* | |
| in | *itemName* | |

**Returns**

The *path*

Definition at line 281 of file Arn.cpp.

#### 10.7.2.21 void ArnM::setConsoleError ( bool *isConsoleError* ) [static]

Definition at line 902 of file Arn.cpp.

#### 10.7.2.22 void ArnM::setDefaultIgnoreSameValue ( bool *isIgnore* = true ) [static]

Set system default skipping of equal value.

**Parameters**

| | | |
|---|---|---|
| in | *isIgnore* | If true, assignment of equal value don't give a changed signal. |

Definition at line 908 of file Arn.cpp.

#### 10.7.2.23 void ArnM::setupErrorlog ( QObject * *errLog* ) [static],[slot]

Definition at line 803 of file Arn.cpp.

#### 10.7.2.24 void ArnM::setValue ( const QString & *path*, int *value* ) [static]

Assign an *integer* to an *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *value* | to be assigned |

Definition at line 394 of file Arn.cpp.

#### 10.7.2.25 void ArnM::setValue ( const QString & *path*, double *value* ) [static]

Assign a *double* to an *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *value* | to be assigned |

Definition at line 405 of file Arn.cpp.

#### 10.7.2.26 void ArnM::setValue ( const QString & *path*, const QString & *value* ) [static]

Assign a *QString* to an *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *value* | to be assigned |

Definition at line 383 of file Arn.cpp.

#### 10.7.2.27 void ArnM::setValue ( const QString & *path*, const QByteArray & *value* ) [static]

Assign a *QByteArray* to an *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *value* | to be assigned |

Definition at line 416 of file Arn.cpp.

#### 10.7.2.28 void ArnM::setValue ( const QString & *path*, const QVariant & *value* ) [static]

Assign a *QVariant* to an *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *value* | to be assigned |

Definition at line 427 of file Arn.cpp.

#### 10.7.2.29 QString ArnM::twinPath ( const QString & *path* ) [static]

Get the bidirectional twin to a given *path*

Example: *path* = "//Measure/depth/value!" ==> return = "//Measure/depth/value"

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The twin *path*

**See also**

Bidirectional Arn Data Objects

Definition at line 336 of file Arn.cpp.

#### 10.7.2.30 QByteArray ArnM::valueByteArray ( const QString & *path* ) [static]

Get the value of *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The *Arn Data Object* as a *QByteArray*

Definition at line 143 of file Arn.cpp.

#### 10.7.2.31 double ArnM::valueDouble ( const QString & *path* ) [static]

Get the value of *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The *Arn Data Object* as a *double*

Definition at line 121 of file Arn.cpp.

#### 10.7.2.32 int ArnM::valueInt ( const QString & *path* ) [static]

Get the value of *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The *Arn Data Object* as an *integer*

Definition at line 110 of file Arn.cpp.

#### 10.7.2.33 QString ArnM::valueString ( const QString & *path* ) [static]

Get the value of *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The *Arn Data Object* as a *QString*

Definition at line 132 of file Arn.cpp.

#### 10.7.2.34 QVariant ArnM::valueVariant ( const QString & *path* ) [static]

Get the value of *Arn Data Object* at *path*

**Parameters**

| | | |
|---|---|---|
| in | *path* | |

**Returns**

The *Arn Data Object* as a *QVariant*

Definition at line 154 of file Arn.cpp.

### 10.7.3 Friends And Related Function Documentation

#### 10.7.3.1 friend class ArnItem [friend]

Definition at line 104 of file Arn.hpp.

The documentation for this class was generated from the following files:

- src/Arn.hpp (1.0.0)
- src/Arn.cpp (1.0.0)

## 10.8 ArnMonitor Class Reference

A client remote monitor to detect changes at server.

`#include <ArnMonitor.hpp>`

### Public Slots

- void foundChildDeleted (QString path)

  *Help telling the monitor about deletion of a previous found child.*

### Signals

- void arnItemCreated (QString path)

  *Signal emitted when an Arn Data Object is created in the tree below.*
- void arnChildFound (QString path)

  *Signal emitted for present and newly created childs in the monitor folder.*
- void arnChildFoundFolder (QString path)

  *Signal emitted for present and newly created folder childs in the monitor folder.*
- void arnChildFoundLeaf (QString path)

  *Signal emitted for present and newly created leaf childs in the monitor folder.*

### Public Member Functions

- ArnMonitor (QObject *parent=0)
- void setClient (ArnClient *client, QString id=QString())

  *Set the client to be used.*
- QString clientId () const

  *Get the id name of the used client*
- void setMonitorPath (QString path, ArnClient *client=0)

  *Set the path to be monitored.*
- QString monitorPath () const

  *Get the monitored path*
- void reStart ()

  *The monitor is restarted.*
- void setReference (void *reference)

  *Set an associated external reference.*
- void * reference () const

  *Get the stored external reference.*

### Protected Attributes

- ArnClient * _arnClient
- QString _monitorPath

### 10.8.1 Detailed Description

A client remote monitor to detect changes at server.

The monitor must be set at a shared path.

When the monitor is started, all the *arnChildFound* signals are emmited for present childs. Later the signals are emmited for newly created childs.

**Example usage**

```
// In class declare
ArnMonitor*  _arnMon;
ArnClient*  _client;

// In class code
_arnMon = new ArnMonitor( this);
_arnMon->setMonitorPath("//Pipes/", _client);
connect( _arnMon, SIGNAL(arnChildFound(QString)), this, SLOT(
  netChildFound(QString)));
```

Definition at line 63 of file ArnMonitor.hpp.

### 10.8.2 Constructor & Destructor Documentation

#### 10.8.2.1 ArnMonitor::ArnMonitor ( QObject * *parent* = 0 ) [explicit]

Definition at line 39 of file ArnMonitor.cpp.

### 10.8.3 Member Function Documentation

#### 10.8.3.1 void ArnMonitor::arnChildFound ( QString *path* ) [signal]

Signal emitted for present and newly created childs in the monitor folder.

The ArnMonitor monitors a folder. Present and newly created objects in this folder will give this signal. For newly created childs, the origin comes from the arnItemCreated() signal, so only non folder objects will then give this signal.

Example: monitorPath = "//Sensors/", created object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/-Temp1/"

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the child |

**See also**

arnItemCreated()

#### 10.8.3.2 void ArnMonitor::arnChildFoundFolder ( QString *path* ) [signal]

Signal emitted for present and newly created folder childs in the monitor folder.

The ArnMonitor monitors a folder. Present and newly created folder objects in this folder will give this signal. For newly created childs, the origin comes from the arnItemCreated() signal, so only non folder objects will then give this signal.

Example: monitorPath = "//Sensors/", created object = "//Sensors/Temp1/value" ==> path to child = "//Sensors/-Temp1/"

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the child |

**See also**

arnItemCreated()
arnChildFound()

#### 10.8.3.3 void ArnMonitor::arnChildFoundLeaf ( QString *path* ) [signal]

Signal emitted for present and newly created leaf childs in the monitor folder.

The ArnMonitor monitors a folder. Present and newly created leaf objects in this folder will give this signal.

Example: monitorPath = "//Sensors/", created object = "//Sensors/count" ==> path to child = "//Sensors/count"

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the child |

**See also**

arnChildFound()

#### 10.8.3.4 void ArnMonitor::arnItemCreated ( QString *path* ) [signal]

Signal emitted when an *Arn Data Object* is created in the tree below.

The ArnMonitor monitors a folder. Created objects in this folder or its children below will give this signal. Only created non folder objects will give this signal.

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the created *Arn Data Object* |

#### 10.8.3.5 QString ArnMonitor::clientId ( ) const

Get the id name of the used *client*

**Returns**

The *client* id name

**See also**

setClient()

Definition at line 55 of file ArnMonitor.cpp.

#### 10.8.3.6 void ArnMonitor::foundChildDeleted ( QString *path* ) [slot]

Help telling the monitor about deletion of a previous found child.

The monitor remembers every child it has signalled. If a deleted child reappears later it will not give a signal unless this function is used.

**Parameters**

| | | |
|---|---|---|
| in | *path* | to the deleted child |

Definition at line 147 of file ArnMonitor.cpp.

#### 10.8.3.7 QString ArnMonitor::monitorPath ( ) const [inline]

Get the monitored *path*

**Returns**

The *path*

**See also**

setMonitorPath()

Definition at line 94 of file ArnMonitor.hpp.

#### 10.8.3.8 void* ArnMonitor::reference ( ) const [inline]

Get the stored external reference.

**Returns**

The associated external reference

**See also**

setReference()

Definition at line 115 of file ArnMonitor.hpp.

#### 10.8.3.9 void ArnMonitor::reStart ( )

The monitor is restarted.

This makes the monitor forget the signals sent for present children and the *arnChildFound* signals are emmited again for present childs.

Definition at line 102 of file ArnMonitor.cpp.

#### 10.8.3.10 void ArnMonitor::setClient ( ArnClient * *client*, QString *id* = QString() )

Set the *client* to be used.

**Parameters**

| | | |
|---|---|---|
| in | *client* | |
| in | *id* | is an optional name to assign to the client. |

Definition at line 47 of file ArnMonitor.cpp.

#### 10.8.3.11 void ArnMonitor::setMonitorPath ( QString *path,* ArnClient * *client* = 0 )

Set the *path* to be monitored.

The monitor must be set at a shared *path*. This function also starts the monitoring.

**Parameters**

| | | |
|---|---|---|
| in | *path* | |
| in | *client* | to be used. If 0, keep previous set client. |

Definition at line 62 of file ArnMonitor.cpp.

#### 10.8.3.12 void ArnMonitor::setReference ( void * *reference* ) [inline]

Set an associated external reference.

This is typically used when having many *ArnMonitors* signal connected to a common slot. The slot can then discover the signalling ArnMonitor:s associated structure for further processing.

**Parameters**

| | | |
|---|---|---|
| in | *reference* | Any external structure or id. |

**See also**

reference()

Definition at line 109 of file ArnMonitor.hpp.

### 10.8.4 Member Data Documentation

#### 10.8.4.1 ArnClient* ArnMonitor::_arnClient [protected]

Definition at line 173 of file ArnMonitor.hpp.

#### 10.8.4.2 QString ArnMonitor::_monitorPath [protected]

Definition at line 174 of file ArnMonitor.hpp.

The documentation for this class was generated from the following files:

- src/ArnMonitor.hpp (1.0.0)
- src/ArnMonitor.cpp (1.0.0)

## 10.9 ArnPersist Class Reference

```
#include <ArnPersist.hpp>
```

### Public Slots

- bool doArchive (QString name=QString())

### Public Member Functions

- ArnPersist (QObject *parent=0)
- ~ArnPersist ()
- bool setMountPoint (const QString &path)

  *Set the persistent enabled tree path.*
- void setPersistDir (const QString &path)

  *Set the VCS persistent file directory root*
- void setArchiveDir (const QString &path)

  *Set the persistent database backup directory.*
- void setVcs (ArnVcs *vcs)

  *Set the Version Control System to be used.*
- bool setupDataBase (QString dbName="persist.db")

  *Setup the persistent database.*

## 10.9.1 Detailed Description

Class for handling persistent *Arn Data object*.

About Persistent Arn Data Object

This class is used at an *ArnServer* to implemennt persistent objects.

**Example usage**

```
// In class declare
ArnPersist  *_persist;
VcsGit  *_git;

// In class code
_persist = new ArnPersist( this);
_persist->setupDataBase("persist.db");
_persist->setArchiveDir("archive");  // Use this directory for
   backup
_persist->setPersistDir("persist");  // use this directory for
   VCS persist files
_persist->setMountPoint("/");
_persist->setVcs( _git);
```

Definition at line 149 of file ArnPersist.hpp.

## 10.9.2 Constructor & Destructor Documentation

### 10.9.2.1 ArnPersist::ArnPersist ( QObject * *parent* = 0 ) [explicit]

Definition at line 150 of file ArnPersist.cpp.

### 10.9.2.2 ArnPersist::~ArnPersist ( )

Definition at line 166 of file ArnPersist.cpp.

## 10.9.3 Member Function Documentation

### 10.9.3.1 bool ArnPersist::doArchive ( QString *name* = QString() ) [slot]

Do a persistent database backup

By default the backup file will be marked by date and clock. Optionally a custom name can be set for the backup file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | is the file name of the backup. QString() is default name. |

**See also**

setArchiveDir()

Definition at line 550 of file ArnPersist.cpp.

#### 10.9.3.2 void ArnPersist::setArchiveDir ( const QString & *path* )

Set the persistent database backup directory.

In this directory, all backup files are stored.

**Parameters**

| | | |
|---|---|---|
| in | *path* | is the persistent file directory *root*. |

**See also**

doArchive()
Persistent Arn Data Objects

Definition at line 179 of file ArnPersist.cpp.

#### 10.9.3.3 bool ArnPersist::setMountPoint ( const QString & *path* )

Set the persistent enabled tree path.

Mountpoint is a folder. When an *Arn Data Object* change to *Save* mode in this folder or anywhere below in the tree, it will be treated as a persistent object.

**Parameters**

| | | |
|---|---|---|
| in | *path* | is the persistent enabled tree. |

**Return values**

| | |
|---|---|
| *false* | if error. |

**See also**

Persistent Arn Data Objects

Definition at line 348 of file ArnPersist.cpp.

#### 10.9.3.4 void ArnPersist::setPersistDir ( const QString & *path* )

Set the VCS persistent file directory *root*

In this directory and below, all VCS persistent files are stored. The *path* correspond to the *root* in Arn.

Example: *path* is set to "/usr/local/arn_persist". There is a file stored at "/usr/local/arn_persist/@/doc/help.html". This file will be mapped to Arn at "//doc/help.html".

**Parameters**

| | | |
|---|---|---|
| in | *path* | is the persistent file directory *root*. |

**See also**

setVcs()
Persistent Arn Data Objects

Definition at line 173 of file ArnPersist.cpp.

#### 10.9.3.5 bool ArnPersist::setupDataBase ( QString *dbName* = "persist.db" )

Setup the persistent database.

Starting a SQLite database to store persistent *Arn Data Object* in.

**Parameters**

| | | |
|---|---|---|
| in | *dbName* | is the name (and path) of the SQLite database file. |

**See also**

Persistent Arn Data Objects

Definition at line 378 of file ArnPersist.cpp.

#### 10.9.3.6 void ArnPersist::setVcs ( ArnVcs * *vcs* )

Set the *Version Control System* to be used.

The VCS is implemented in a class derived from ArnVcs.

**Parameters**

| | | |
|---|---|---|
| in | *vcs* | is the class implementing the VCS. |

**See also**

setPersistDir()
Persistent Arn Data Objects

Definition at line 185 of file ArnPersist.cpp.

The documentation for this class was generated from the following files:

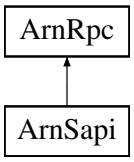
- src/ArnPersist.hpp (1.0.0)
- src/ArnPersist.cpp (1.0.0)

## 10.10 ArnRpc Class Reference

Remote Procedure Call.

`#include <ArnRpc.hpp>`

Inheritance diagram for ArnRpc:




## Classes

- struct Mode

## Public Slots

- void sendText (QString txt)

  *Send a general text message to the other end of the used pipe*

## Signals

- void pipeClosed ()

  *Signal emitted when the used pipe is closed.*
- void textReceived (QString text)

  *Signal emitted when a general text message is received.*

## Public Member Functions

- ArnRpc (QObject ∗parent=0)
- QString pipePath () const

  *Get the path for the used pipe*
- bool open (QString pipePath)
- void setPipe (ArnItem ∗pipe)
- void setReceiver (QObject ∗receiver)
- void setMethodPrefix (QString prefix)
- void setIncludeSender (bool v)
- void setMode (Mode mode)
- Mode mode () const

  *Get the mode.*
- void addSenderSignals (QObject ∗sender, QString prefix)
- bool invoke (const QString &funcName, MQGenericArgument val0=MQGenericArgument(0), MQGenericArgument val1=MQGenericArgument(), MQGenericArgument val2=MQGenericArgument(), MQGenericArgument val3=MQGenericArgument(), MQGenericArgument val4=MQGenericArgument(), MQGenericArgument val5=MQGenericArgument(), MQGenericArgument val6=MQGenericArgument(), MQGenericArgument val7=MQGenericArgument())

  *Calls a named remote procedure.*
- ArnRpc ∗ rpcSender ()
- void batchConnect (const QRegExp &rgx, const QObject ∗receiver, const QString &replace, Mode mode=Mode())

  *Make batch connection from this ArnRpc:s signals to another receivers slots.*
- void batchConnect (const QObject ∗sender, const QRegExp &rgx, const QString &replace, Mode mode=Mode())

  *Make batch connection from one senders signals to this ArnRpc:s slots.*

### Static Public Member Functions

- static ArnRpc ∗ rpcSender (QObject ∗receiver)
- static void batchConnect (const QObject ∗sender, const QRegExp &rgx, const QObject ∗receiver, const QString &replace, Mode mode=Mode())

  *Make batch connection from one senders signals to another receivers slots.*

### Protected Member Functions

- void errorLog (QString errText, ArnError err=ArnError::Undef, void ∗reference=0)

## 10.10.1 Detailed Description

Remote Procedure Call.

About RPC and SAPI

This is the basic funtionality of RPC. It's recommended to use ArnSapi which uses a higher level model. For now the ArnRpc class is more sparsely documented.

**Example usage**

```
    // In class declare  (MyClass)
    ArnRpc*  _rpcCommon;

    // In class code  (MyClass)
    <em>rpcCommon = new ArnRpc( this);
    _rpcCommon->setIncludeSender( true);
    _rpcCommon->setMethodPrefix("rpc</em>");
    _rpcCommon->setReceiver( this);
    _rpcCommon->setMode( ArnRpc::Mode::Provider);
    _rpcCommon->open("//Pipes/pipeCommon");
.
.
void  MyClass::rpc_test( ArnRpc* sender, QByteArray ba, QString str, int
      i)
{
    if (sender) qDebug() << "RPC sender=" << sender->pipePath();
    qDebug() << "RPC-test ba=" << ba << " str=" << str << " int=" << i;
}

void  MyClass::rpc_ver( ArnRpc* sender)
{
    // Reply to requester the version text
    sender->invoke("ver", MQ_ARG( QString, verText, "MySytem
       Version 1.0"));
}
```

Definition at line 112 of file ArnRpc.hpp.

## 10.10.2 Constructor & Destructor Documentation

### 10.10.2.1 ArnRpc::ArnRpc ( QObject ∗ *parent* = 0 ) [explicit]

Definition at line 134 of file ArnRpc.cpp.

## 10.10.3 Member Function Documentation

### 10.10.3.1 void ArnRpc::addSenderSignals ( QObject ∗ *sender,* QString *prefix* )

Definition at line 232 of file ArnRpc.cpp.

#### 10.10.3.2 void ArnRpc::batchConnect ( const QObject * *sender*, const QRegExp & *rgx*, const QObject * *receiver*, const QString & *replace*, Mode *mode* = Mode() ) [static]

Make batch connection from one senders signals to another receivers slots.

Used when there is a pattern in the naming of the signals and slots. It's assumed that naming for slots are unique regardless of its case i.e. using both test() and tesT() are not allowed.

Example: `batchConnect( _commonSapi, QRegExp("^rq_(.+)"), this, "chat\\\\1");`
connects signal: `rq_info(QString,QString)` to slot: `chatInfo(QString,QString)`

**Parameters**

| | | |
|---|---|---|
| in | *sender* | is the sending QObject. |
| in | *rgx* | is the regular expression for selecting signals. |
| in | *receiver* | is the receiving QObject. |
| in | *replace* | is the conversion for naming the slots. |
| in | *mode* | Used modes: *Debug*, *NoDefaultArgs* |

Definition at line 734 of file ArnRpc.cpp.

#### 10.10.3.3 void ArnRpc::batchConnect ( const QRegExp & *rgx*, const QObject * *receiver*, const QString & *replace*, Mode *mode* = Mode() ) [inline]

Make batch connection from this ArnRpc:s signals to another receivers slots.

Example: `_commonSapi.batchConnect( QRegExp("^rq_(.+)"), this, "chat\\\\1");`
connects signal: `rq_info(QString,QString)` to slot: `chatInfo(QString,QString)`

**Parameters**

| | | |
|---|---|---|
| in | *rgx* | is the regular expression for selecting signals. |
| in | *receiver* | is the receiving QObject. |
| in | *replace* | is the conversion for naming the slots. |
| in | *mode* | |

**See also**

batchConnect(const QObject*, const QRegExp&, const QObject*, const QString&, Mode)

Definition at line 211 of file ArnRpc.hpp.

#### 10.10.3.4 void ArnRpc::batchConnect ( const QObject * *sender*, const QRegExp & *rgx*, const QString & *replace*, Mode *mode* = Mode() ) [inline]

Make batch connection from one senders signals to this ArnRpc:s slots.

Example: `_commonSapi.batchConnect( _commonSapi, QRegExp("^rq_(.+)"), "chat\\\\1");`
connects signal: `rq_info(QString,QString)` to slot: `chatInfo(QString,QString)`

**Parameters**

| | | |
|---|---|---|
| in | *sender* | is the sending QObject. |
| in | *rgx* | is the regular expression for selecting signals. |
| in | *replace* | is the conversion for naming the slots. |
| in | *mode* | |

**See also**

batchConnect(const QObject*, const QRegExp&, const QObject*, const QString&, Mode)

Definition at line 228 of file ArnRpc.hpp.

#### 10.10.3.5 void ArnRpc::errorLog ( QString *errText*, ArnError *err* = ArnError::Undef, void * *reference* = 0 ) [protected]

Definition at line 724 of file ArnRpc.cpp.

#### 10.10.3.6 bool ArnRpc::invoke ( const QString & *funcName*, MQGenericArgument *val0* = MQGenericArgument(0), MQGenericArgument *val1* = MQGenericArgument(), MQGenericArgument *val2* = MQGenericArgument(), MQGenericArgument *val3* = MQGenericArgument(), MQGenericArgument *val4* = MQGenericArgument(), MQGenericArgument *val5* = MQGenericArgument(), MQGenericArgument *val6* = MQGenericArgument(), MQGenericArgument *val7* = MQGenericArgument() )

Calls a named remote procedure.

This is the low level way to call a remote procedure. It can freely call anything without declaring it. For high level calls use ArnSapi.

This function works similar to QMetaObject::invokeMethod(). The called name is prefixed before the final call is made. Using the label in MQ_ARG() makes dubugging easier, as the parameter is named.

Example: `rpc->invoke("myfunc", MQ_ARG( QString, mypar, "Test XYZ"));`

**Parameters**

| | | |
|---|---|---|
| in | *funcName* | is the name of the called procedure. |
| in | *val0* | first arg. |
| in | *val1* | second arg. |

Definition at line 278 of file ArnRpc.cpp.

#### 10.10.3.7 ArnRpc::Mode ArnRpc::mode ( ) const

Get the mode.

**Returns**

current *mode*

Definition at line 226 of file ArnRpc.cpp.

#### 10.10.3.8 bool ArnRpc::open ( QString *pipePath* )

Definition at line 154 of file ArnRpc.cpp.

#### 10.10.3.9 void ArnRpc::pipeClosed ( ) [signal]

Signal emitted when the used pipe is closed.

The *pipe* closes when its *Arn Data Object* is destroyed, i.e. the session is considered ended.

#### 10.10.3.10 QString ArnRpc::pipePath ( ) const

Get the path for the used *pipe*

**Return values**

| | |
|---|---|
| *false* | if error |

**See also**

Bidirectional Arn Data Objects

Definition at line 146 of file ArnRpc.cpp.

#### 10.10.3.11 ArnRpc * ArnRpc::rpcSender ( )

Definition at line 259 of file ArnRpc.cpp.

#### 10.10.3.12 ArnRpc * ArnRpc::rpcSender ( QObject * *receiver* ) [static]

Definition at line 267 of file ArnRpc.cpp.

#### 10.10.3.13 void ArnRpc::sendText ( QString *txt* ) [slot]

Send a general text message to the other end of the used *pipe*

Is used by ArnRpc to give errors and help messages, mostly for debugging.

**Parameters**

| | | |
|---|---|---|
| in | *txt* | is the text to be sent |

**See also**

textReceived();

Definition at line 718 of file ArnRpc.cpp.

#### 10.10.3.14 void ArnRpc::setIncludeSender ( bool *v* )

Definition at line 214 of file ArnRpc.cpp.

#### 10.10.3.15 void ArnRpc::setMethodPrefix ( QString *prefix* )

Definition at line 208 of file ArnRpc.cpp.

#### 10.10.3.16 void ArnRpc::setMode ( Mode *mode* )

Definition at line 220 of file ArnRpc.cpp.

#### 10.10.3.17 void ArnRpc::setPipe ( ArnItem * *pipe* )

Definition at line 180 of file ArnRpc.cpp.

#### 10.10.3.18 void ArnRpc::setReceiver ( QObject * *receiver* )

Definition at line 196 of file ArnRpc.cpp.

#### 10.10.3.19 void ArnRpc::textReceived ( QString *text* ) [signal]

Signal emitted when a general text message is received.

The text message is received from the other end of the used *pipe*.

**Parameters**

| | | |
|---|---|---|
| in | *text* | is the received text |

**See also**

sendText();

The documentation for this class was generated from the following files:

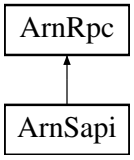
- src/ArnRpc.hpp (1.0.0)
- src/ArnRpc.cpp (1.0.0)

## 10.11 ArnSapi Class Reference

Service API.

`#include <ArnSapi.hpp>`

Inheritance diagram for ArnSapi:




### Public Member Functions

- ArnSapi (QObject *parent=0)
- bool open (QString pipePath, Mode mode=Mode(), const char *providerPrefix=0, const char *requesterPrefix=0)

  *Open a new Service API.*

### Additional Inherited Members

### 10.11.1 Detailed Description

Service API.

About RPC and SAPI

This class serves as a base class for *Service Application Programming Interface*. It should be derived to a custom class that descibe a specific *SAPI*.

By default all *provider* services are prefixed by "pv_" and all *requester* "services" are prefixed by "rq_". This standard can be changed.

**Example usage**

```
class ChatSapi : public ArnSapi
{
    Q_OBJECT
public:
    explicit ChatSapi( QObject* parent = 0) : ArnSapi( parent)  {}

signals:
MQ_PUBLIC_ACCESS
    void  pv_list();
    void  pv_newMsg( QString name, QString msg);
    void  pv_infoQ();

    void  rq_updateMsg( int seq, QString name, QString msg);
    void  rq_info( QString name, QString ver);
};

    // In class declare  (MyClass)
    ChatSapi*  _commonSapi;

    // In class code  (MyClass)
    <em>commonSapi = new ChatSapi( this);
    _commonSapi->open("//Chat/Pipes/pipeCommon!", ArnSapi::Mode::Provider
      );
    _commonSapi->batchConnect( QRegExp("^pv</em>(.+)"), this, "chat\\1");
.
.
void  ServerMain::chatNewMsg( QString name, QString msg)
{
    int  seq = ...;
    _commonSapi->rq_updateMsg( seq, name, msg);
}

void  MyClass::chatInfoQ()
{
    ChatSapi*  sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    sapi->rq_info("Arn Chat Demo", "1.0");
}
```

**Examples:**

ChatSapi.hpp.

Definition at line 101 of file ArnSapi.hpp.

### 10.11.2 Constructor & Destructor Documentation

#### 10.11.2.1 ArnSapi::ArnSapi ( QObject * *parent* = 0 ) [explicit]

**Examples:**

ChatSapi.hpp.

Definition at line 37 of file ArnSapi.cpp.

### 10.11.3 Member Function Documentation

#### 10.11.3.1 bool ArnSapi::open ( QString *pipePath*, Mode *mode* = Mode (), const char * *providerPrefix* = 0, const char * *requesterPrefix* = 0 )

Open a new Service API.

The opened Sapi can be either the *provider* side or the *requester* side, which is indicated by *mode*.

Typically the *provider* is only using *mode Provider*. The *requester* can use default *mode* for a static *pipe* and typically use the *UuidAutoDestroy mode* for dynamic session *pipes*.

**Parameters**

| | | |
|---|---|---|
| in | *pipePath* | is the path used for Sapi |
| in | *mode* | |
| in | *providerPrefix* | to set a custom prefix for *provider* signals. |
| in | *requesterPrefix* | to set a custom prefix for *requester* signals. |

**Return values**

| | |
|---|---|
| *false* | if error |

**See also**

Pipes

Definition at line 43 of file ArnSapi.cpp.

The documentation for this class was generated from the following files:

- src/ArnSapi.hpp (1.0.0)
- src/ArnSapi.cpp (1.0.0)

## 10.12 ArnScript Class Reference

`#include <ArnScript.hpp>`

### Signals

- void errorText (QString txt)

### Public Member Functions

- ArnScript (QObject *parent=0)
- QScriptEngine & engine () const
- bool evaluate (QByteArray script, QString idName)
- bool evaluateFile (QString fileName)
- bool logUncaughtError (QScriptValue &scriptValue)
- QString idName () const
- virtual ArnClient * getClient (QString clientId)

### Protected Member Functions

- void errorLog (QString errText, ArnError err=ArnError::Undef, void *reference=0)

### Static Protected Member Functions

- static QScriptValue printFunction (QScriptContext *context, QScriptEngine *engine)

### Protected Attributes

- QScriptEngine * _engine
- ArnItemProto * _itemProto
- ArnMonitorProto * _monitorProto
- ArnDepOfferProto * _depOfferProto
- ArnDepProto * _depProto

### 10.12.1 Detailed Description

Definition at line 179 of file ArnScript.hpp.

### 10.12.2 Constructor & Destructor Documentation

#### 10.12.2.1 ArnScript::ArnScript ( QObject * *parent* = 0 ) [explicit]

Definition at line 48 of file ArnScript.cpp.

### 10.12.3 Member Function Documentation

#### 10.12.3.1 QScriptEngine& ArnScript::engine ( ) const [inline]

Definition at line 184 of file ArnScript.hpp.

#### 10.12.3.2 void ArnScript::errorLog ( QString *errText,* ArnError *err* = ArnError::Undef, void * *reference* = 0 ) [protected]

Definition at line 159 of file ArnScript.cpp.

#### 10.12.3.3 void ArnScript::errorText ( QString *txt* ) [signal]

#### 10.12.3.4 bool ArnScript::evaluate ( QByteArray *script,* QString *idName* )

Definition at line 97 of file ArnScript.cpp.

#### 10.12.3.5 bool ArnScript::evaluateFile ( QString *fileName* )

Definition at line 108 of file ArnScript.cpp.

#### 10.12.3.6 ArnClient * ArnScript::getClient ( QString *clientId* ) [virtual]

Definition at line 168 of file ArnScript.cpp.

#### 10.12.3.7 QString ArnScript::idName ( ) const [inline]

Definition at line 188 of file ArnScript.hpp.

#### 10.12.3.8 bool ArnScript::logUncaughtError ( QScriptValue & *scriptValue* )

Definition at line 117 of file ArnScript.cpp.

#### 10.12.3.9 QScriptValue ArnScript::printFunction ( QScriptContext * *context,* QScriptEngine * *engine* ) [static], [protected]

Definition at line 141 of file ArnScript.cpp.

### 10.12.4 Member Data Documentation

#### 10.12.4.1 ArnDepOfferProto* ArnScript::_depOfferProto [protected]

Definition at line 208 of file ArnScript.hpp.

#### 10.12.4.2 ArnDepProto* ArnScript::_depProto [protected]

Definition at line 209 of file ArnScript.hpp.

#### 10.12.4.3 QScriptEngine* ArnScript::_engine [protected]

Definition at line 205 of file ArnScript.hpp.

#### 10.12.4.4 ArnItemProto* ArnScript::_itemProto [protected]

Definition at line 206 of file ArnScript.hpp.

#### 10.12.4.5 ArnMonitorProto* ArnScript::_monitorProto [protected]

Definition at line 207 of file ArnScript.hpp.

The documentation for this class was generated from the following files:

- src/ArnScript.hpp (1.0.0)
- src/ArnScript.cpp (1.0.0)

## 10.13 ArnScriptJob Class Reference

`#include <ArnScriptJob.hpp>`

### Public Slots

- void setWatchDogTime (int time)
- void yield ()
- void quit ()
- void errorLog (QString txt)

### Signals

- void sigQuit ()

### Public Member Functions

- ArnScriptJob (int id, QObject *parent=0)

### Properties

- bool sleepState
- int watchDog
- int poll
- QString name

## 10.13.1 Detailed Description

Interface class to be normally used, is also Script Job interface

Definition at line 134 of file ArnScriptJob.hpp.

## 10.13.2 Constructor & Destructor Documentation

### 10.13.2.1 ArnScriptJob::ArnScriptJob ( int *id,* QObject * *parent* = 0 ) [explicit]

Definition at line 374 of file ArnScriptJob.cpp.

## 10.13.3 Member Function Documentation

### 10.13.3.1 void ArnScriptJob::errorLog ( QString *txt* ) [inline], [slot]

Definition at line 151 of file ArnScriptJob.hpp.

### 10.13.3.2 void ArnScriptJob::quit ( ) [inline], [slot]

Definition at line 150 of file ArnScriptJob.hpp.

### 10.13.3.3 void ArnScriptJob::setWatchDogTime ( int *time* ) [inline], [slot]

Definition at line 148 of file ArnScriptJob.hpp.

### 10.13.3.4 void ArnScriptJob::sigQuit ( ) [signal]

### 10.13.3.5 void ArnScriptJob::yield ( ) [inline], [slot]

Definition at line 149 of file ArnScriptJob.hpp.

## 10.13.4 Property Documentation

### 10.13.4.1 QString ArnScriptJob::name [read]

Definition at line 140 of file ArnScriptJob.hpp.

### 10.13.4.2 int ArnScriptJob::poll [read], [write]

Definition at line 139 of file ArnScriptJob.hpp.

#### 10.13.4.3 bool ArnScriptJob::sleepState [read],[write]

Definition at line 137 of file ArnScriptJob.hpp.

#### 10.13.4.4 int ArnScriptJob::watchDog [read],[write]

Definition at line 138 of file ArnScriptJob.hpp.

The documentation for this class was generated from the following files:

- src/ArnScriptJob.hpp (1.0.0)
- src/ArnScriptJob.cpp (1.0.0)

## 10.14 ArnScriptJobControl Class Reference

Is thread-safe (except doSetupJob)

`#include <ArnScriptJob.hpp>`

### Public Slots

- void setScript (QByteArray script)

### Signals

- void scriptChanged (int id)
- void errorText (QString txt)

### Public Member Functions

- ArnScriptJobControl (QObject *parent=0)
- int id ()
- QString name () const
- void setName (QString name)
- void addInterface (QString id)
- void addInterfaceList (QStringList interfaceList)
- QByteArray script () const
- void loadScriptFile (QString fileName)
- QVariant config (const char *name) const
- bool setConfig (const char *name, const QVariant &value)
- void addConfig (QObject *obj)
- void setThreaded (bool isThreaded)
- void doSetupJob (ArnScriptJob *job, ArnScriptJobFactory *jobFactory)

  *Not threadsafe, only run in same thread as script.*

### 10.14.1 Detailed Description

Is thread-safe (except doSetupJob)

Definition at line 172 of file ArnScriptJob.hpp.

### 10.14.2 Constructor & Destructor Documentation

#### 10.14.2.1 ArnScriptJobControl::ArnScriptJobControl ( QObject * *parent* = 0 ) [explicit]

Definition at line 385 of file ArnScriptJob.cpp.

### 10.14.3 Member Function Documentation

#### 10.14.3.1 void ArnScriptJobControl::addConfig ( QObject * *obj* )

Definition at line 484 of file ArnScriptJob.cpp.

#### 10.14.3.2 void ArnScriptJobControl::addInterface ( QString *id* )

Definition at line 422 of file ArnScriptJob.cpp.

#### 10.14.3.3 void ArnScriptJobControl::addInterfaceList ( QStringList *interfaceList* )

Definition at line 431 of file ArnScriptJob.cpp.

#### 10.14.3.4 QVariant ArnScriptJobControl::config ( const char * *name* ) const

Definition at line 512 of file ArnScriptJob.cpp.

#### 10.14.3.5 void ArnScriptJobControl::doSetupJob ( ArnScriptJob * *job*, ArnScriptJobFactory * *jobFactory* )

Not threadsafe, only run in same thread as script.
Definition at line 496 of file ArnScriptJob.cpp.

#### 10.14.3.6 void ArnScriptJobControl::errorText ( QString *txt* ) [signal]

#### 10.14.3.7 int ArnScriptJobControl::id ( )

Definition at line 402 of file ArnScriptJob.cpp.

#### 10.14.3.8 void ArnScriptJobControl::loadScriptFile ( QString *fileName* )

Definition at line 460 of file ArnScriptJob.cpp.

#### 10.14.3.9 QString ArnScriptJobControl::name ( ) const

Definition at line 412 of file ArnScriptJob.cpp.

#### 10.14.3.10 QByteArray ArnScriptJobControl::script ( ) const

Definition at line 450 of file ArnScriptJob.cpp.

#### 10.14.3.11 void ArnScriptJobControl::scriptChanged ( int *id* ) [signal]

#### 10.14.3.12 bool ArnScriptJobControl::setConfig ( const char * *name,* const QVariant & *value* )

Definition at line 472 of file ArnScriptJob.cpp.

#### 10.14.3.13 void ArnScriptJobControl::setName ( QString *name* )

Definition at line 394 of file ArnScriptJob.cpp.

#### 10.14.3.14 void ArnScriptJobControl::setScript ( QByteArray *script* ) [slot]

Definition at line 440 of file ArnScriptJob.cpp.

#### 10.14.3.15 void ArnScriptJobControl::setThreaded ( bool *isThreaded* ) [inline]

Definition at line 188 of file ArnScriptJob.hpp.

The documentation for this class was generated from the following files:

- src/ArnScriptJob.hpp (1.0.0)
- src/ArnScriptJob.cpp (1.0.0)

## 10.15 ArnScriptJobFactory Class Reference

Must be thread-safe as subclassed.

`#include <ArnScriptJob.hpp>`

### Public Member Functions

- ArnScriptJobFactory ()
- virtual ~ArnScriptJobFactory ()
- virtual bool installExtension (QString id, QScriptEngine &engine, const ArnScriptJobControl *jobControl=0)=0
- virtual ArnClient * getClient (QString id)

### Static Protected Member Functions

- static void setupJsObj (const QString &id, const QScriptValue &jsObj, QScriptEngine &engine)
- static bool setupInterface (const QString &id, QObject *interface, QScriptEngine &engine)

### 10.15.1 Detailed Description

Must be thread-safe as subclassed.

Definition at line 156 of file ArnScriptJob.hpp.

### 10.15.2 Constructor & Destructor Documentation

#### 10.15.2.1 ArnScriptJobFactory::ArnScriptJobFactory ( ) [explicit]

Definition at line 334 of file ArnScriptJob.cpp.

#### 10.15.2.2 ArnScriptJobFactory::~ArnScriptJobFactory ( ) [virtual]

Definition at line 339 of file ArnScriptJob.cpp.

### 10.15.3 Member Function Documentation

#### 10.15.3.1 ArnClient * ArnScriptJobFactory::getClient ( QString *id* ) [virtual]

Definition at line 344 of file ArnScriptJob.cpp.

#### 10.15.3.2 virtual bool ArnScriptJobFactory::installExtension ( QString *id,* QScriptEngine & *engine,* const ArnScriptJobControl * *jobControl* = 0 ) [pure virtual]

#### 10.15.3.3 bool ArnScriptJobFactory::setupInterface ( const QString & *id,* QObject * *interface,* QScriptEngine & *engine* ) [static], [protected]

Definition at line 356 of file ArnScriptJob.cpp.

#### 10.15.3.4 void ArnScriptJobFactory::setupJsObj ( const QString & *id,* const QScriptValue & *jsObj,* QScriptEngine & *engine* ) [static], [protected]

Definition at line 350 of file ArnScriptJob.cpp.

The documentation for this class was generated from the following files:

- src/ArnScriptJob.hpp (1.0.0)
- src/ArnScriptJob.cpp (1.0.0)

## 10.16 ArnScriptJobs Class Reference

`#include <ArnScriptJobs.hpp>`

### Classes

- struct **JobSlot**
- struct Type

### Public Member Functions

- ArnScriptJobs (QObject *parent=0)
- void addJob (ArnScriptJobControl *jobConfig, int prio=1)
- void setFactory (ArnScriptJobFactory *jobFactory)
- void start (Type type=Type::Cooperative)

### 10.16.1 Detailed Description

Definition at line 87 of file ArnScriptJobs.hpp.

### 10.16.2 Constructor & Destructor Documentation

#### 10.16.2.1 ArnScriptJobs::ArnScriptJobs ( QObject * *parent* = 0 ) [explicit]

Definition at line 141 of file ArnScriptJobs.cpp.

### 10.16.3 Member Function Documentation

#### 10.16.3.1 void ArnScriptJobs::addJob ( ArnScriptJobControl * *jobConfig*, int *prio* = 1 )

Definition at line 150 of file ArnScriptJobs.cpp.

#### 10.16.3.2 void ArnScriptJobs::setFactory ( ArnScriptJobFactory * *jobFactory* )

Definition at line 162 of file ArnScriptJobs.cpp.

#### 10.16.3.3 void ArnScriptJobs::start ( Type *type* = Type::Cooperative )

Definition at line 168 of file ArnScriptJobs.cpp.

The documentation for this class was generated from the following files:

- src/ArnScriptJobs.hpp (1.0.0)
- src/ArnScriptJobs.cpp (1.0.0)

## 10.17 ArnServer Class Reference

Class for making an *Arn Server*.

`#include <ArnServer.hpp>`

### Classes

- struct Type

### Public Member Functions

- ArnServer (Type serverType, QObject *parent=0)
  *Create an Arn server object.*
- void start (int port=0)
  *Start the Arn server*

### 10.17.1 Detailed Description

Class for making an *Arn Server*.

About Sharing Arn Data Objects

**Example usage**

```
// In class declare
ArnServer*  _server;

// In class code
```

```
_server = new ArnServer( ArnServer::Type::NetSync
  , this);
_server->start();
```

**Examples:**

ServerMain.cpp, and ServerMain.hpp.

Definition at line 56 of file ArnServer.hpp.

### 10.17.2 Constructor & Destructor Documentation

#### 10.17.2.1 ArnServer::ArnServer ( Type *serverType*, QObject * *parent* = 0 )

Create an Arn *server* object.

**Parameters**

| | | |
|---|---|---|
| in | *serverType* | For now only *NetSync* is available. |

Definition at line 42 of file ArnServer.cpp.

### 10.17.3 Member Function Documentation

#### 10.17.3.1 void ArnServer::start ( int *port* = 0 )

Start the Arn *server*

**Parameters**

| | | |
|---|---|---|
| in | *port* | is the port number (default 2022). |

Definition at line 51 of file ArnServer.cpp.

The documentation for this class was generated from the following files:

- src/ArnServer.hpp (1.0.0)
- src/ArnServer.cpp (1.0.0)

## 10.18 ArnLink::Flags Struct Reference

`#include <ArnLink.hpp>`

### Public Types

- enum E { Folder = 0x01, CreateAllowed = 0x02, SilentError = 0x04, Threaded = 0x08 }

### 10.18.1 Detailed Description

Definition at line 64 of file ArnLink.hpp.

### 10.18.2 Member Enumeration Documentation

#### 10.18.2.1 enum ArnLink::Flags::E

**Enumerator:**

***Folder***

***CreateAllowed***

***SilentError***

***Threaded***

Definition at line 65 of file ArnLink.hpp.

The documentation for this struct was generated from the following file:

- src/ArnLink.hpp (1.0.0)

## 10.19 ArnItem::Mode Struct Reference

General global mode of an *Arn Data Object*

`#include <ArnItem.hpp>`

### Public Types

- enum E { BiDir = 0x01, Pipe = 0x02, Save = 0x04 }

### 10.19.1 Detailed Description

General global mode of an *Arn Data Object*

Definition at line 79 of file ArnItem.hpp.

### 10.19.2 Member Enumeration Documentation

#### 10.19.2.1 enum ArnItem::Mode::E

**Enumerator:**

***BiDir*** A two way object, typically for validation or pipe.

***Pipe*** Implies *BiDir* and all data is preserved as a stream.

***Save*** Data is persistent and will be saved.

Definition at line 80 of file ArnItem.hpp.

The documentation for this struct was generated from the following file:

- src/ArnItem.hpp (1.0.0)

## 10.20 ArnRpc::Mode Struct Reference

`#include <ArnRpc.hpp>`

### Public Types

- enum E {
  Provider = 0x01, AutoDestroy = 0x02, UuidPipe = 0x04, NoDefaultArgs = 0x08,
  Debug = 0x10, UuidAutoDestroy = UuidPipe | AutoDestroy }

#### 10.20.1 Detailed Description

Definition at line 116 of file ArnRpc.hpp.

#### 10.20.2 Member Enumeration Documentation

##### 10.20.2.1 enum ArnRpc::Mode::E

**Enumerator:**

***Provider*** Provider side (opposed to requester)

***AutoDestroy*** Use *AutoDestroy* for the pipe, i.e. it is closed when tcp/ip is broken.

***UuidPipe*** Use an unique uuid in the pipe name.

***NoDefaultArgs*** If guarantied no default arguments, member name overload is ok.

***Debug*** Debug mode, dumping done batch connections.

***UuidAutoDestroy*** Convenience, combined *UuidPipe* and *AutoDestroy*

Definition at line 117 of file ArnRpc.hpp.

The documentation for this struct was generated from the following file:

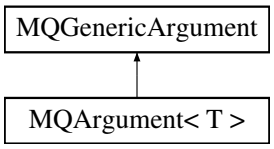
- src/ArnRpc.hpp (1.0.0)

## 10.21 MQArgument< T > Class Template Reference

Similar to QArgument but with added argument label (parameter name)

`#include <ArnRpc.hpp>`

Inheritance diagram for MQArgument< T >:




### Public Member Functions

- MQArgument (const char *aName, const char *aLabel, const T &aData)

#### 10.21.1 Detailed Description

**template<class T>class MQArgument< T >**

Similar to QArgument but with added argument label (parameter name)

Definition at line 70 of file ArnRpc.hpp.

### 10.21.2 Constructor & Destructor Documentation

#### 10.21.2.1 template<class T > MQArgument< T >::MQArgument ( const char * *aName*, const char * *aLabel*, const T & *aData* ) [inline]

Definition at line 73 of file ArnRpc.hpp.

The documentation for this class was generated from the following file:

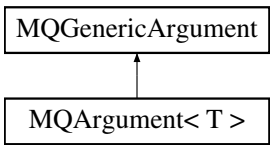
- src/ArnRpc.hpp (1.0.0)

## 10.22 MQGenericArgument Class Reference

Similar to QGenericArgument but with added argument label (parameter name)

`#include <ArnRpc.hpp>`

Inheritance diagram for MQGenericArgument:




### Public Member Functions

- MQGenericArgument (const char *aName=0, const char *aLabel=0, const void *aData=0)
- MQGenericArgument (const QGenericArgument &qgenArg)
- const char * label () const

### 10.22.1 Detailed Description

Similar to QGenericArgument but with added argument label (parameter name)

Definition at line 54 of file ArnRpc.hpp.

### 10.22.2 Constructor & Destructor Documentation

#### 10.22.2.1 MQGenericArgument::MQGenericArgument ( const char * *aName* = 0, const char * *aLabel* = 0, const void * *aData* = 0 ) [inline]

Definition at line 57 of file ArnRpc.hpp.

#### 10.22.2.2 MQGenericArgument::MQGenericArgument ( const QGenericArgument & *qgenArg* ) [inline]

Definition at line 59 of file ArnRpc.hpp.

### 10.22.3 Member Function Documentation

#### 10.22.3.1 const char* MQGenericArgument::label ( ) const [inline]

Definition at line 61 of file ArnRpc.hpp.

The documentation for this class was generated from the following file:

- src/ArnRpc.hpp (1.0.0)

## 10.23 ArnLink::NameF Struct Reference

`#include <ArnLink.hpp>`

### Public Types

- enum E { NoFolderMark = 0x01, EmptyOk = 0x02, Relative = 0x04 }
  *Selects a format for path or item name.*

### 10.23.1 Detailed Description

Definition at line 73 of file ArnLink.hpp.

### 10.23.2 Member Enumeration Documentation

#### 10.23.2.1 enum ArnLink::NameF::E

Selects a format for path or item name.

**Enumerator:**

***NoFolderMark*** Only on discrete names, no effect on path. "test/" ==> "test".

***EmptyOk*** Path: "/@/test" ==> "//test", Item: "@" ==> "".

***Relative*** Only on path, no effect on discrete names. "/test/value" ==> "test/value".

Definition at line 75 of file ArnLink.hpp.

The documentation for this struct was generated from the following file:

- src/ArnLink.hpp (1.0.0)

## 10.24 ArnError::StdCode Struct Reference

`#include <ArnError.hpp>`

### Public Types

- enum E {
  Ok = 0, Info = 1, Warning = 2, Err_Undef = 15,
  Err_Custom = 16 }

### 10.24.1 Detailed Description

Definition at line 41 of file ArnError.hpp.

### 10.24.2 Member Enumeration Documentation

#### 10.24.2.1 enum ArnError::StdCode::E

**Enumerator:**

***Ok***

***Info***

***Warning***

***Err_Undef***

***Err_Custom***

Definition at line 43 of file ArnError.hpp.

The documentation for this struct was generated from the following file:

- src/ArnError.hpp (1.0.0)

## 10.25 ArnItem::SyncMode Struct Reference

The client session sync mode of an *Arn Data Object*

`#include <ArnItem.hpp>`

### Public Types

- enum E { Normal = 0x000, Monitor = 0x001, Master = 0x100, AutoDestroy = 0x200 }

### 10.25.1 Detailed Description

The client session sync mode of an *Arn Data Object*

Definition at line 91 of file ArnItem.hpp.

### 10.25.2 Member Enumeration Documentation

#### 10.25.2.1 enum ArnItem::SyncMode::E

**Enumerator:**

***Normal*** default

***Monitor*** Monitor of server object for client.

***Master*** The client is default generator of data.

***AutoDestroy*** Destroy this *Arn Data Object* when client (tcp/ip) closes.

Definition at line 92 of file ArnItem.hpp.

The documentation for this struct was generated from the following file:

- src/ArnItem.hpp (1.0.0)

## 10.26 ArnLink::Type Struct Reference

`#include <ArnLink.hpp>`

### Public Types

- enum E {
  Null = 0, Int = 1, Double = 2, ByteArray = 3,
  String = 4, Variant = 5 }

### 10.26.1 Detailed Description

Definition at line 53 of file ArnLink.hpp.

### 10.26.2 Member Enumeration Documentation

#### 10.26.2.1 enum ArnLink::Type::E

**Enumerator:**

***Null***

***Int***

***Double***

***ByteArray***

***String***

***Variant***

Definition at line 54 of file ArnLink.hpp.

The documentation for this struct was generated from the following file:

- src/ArnLink.hpp (1.0.0)

## 10.27 ArnScriptJobs::Type Struct Reference

`#include <ArnScriptJobs.hpp>`

### Public Types

- enum E { Null, Cooperative, Preemptive }

### 10.27.1 Detailed Description

Definition at line 91 of file ArnScriptJobs.hpp.

### 10.27.2 Member Enumeration Documentation

#### 10.27.2.1 enum ArnScriptJobs::Type::E

**Enumerator:**

***Null***

***Cooperative***

***Preemptive***

Definition at line 92 of file ArnScriptJobs.hpp.

The documentation for this struct was generated from the following file:

- src/ArnScriptJobs.hpp (1.0.0)

## 10.28 ArnServer::Type Struct Reference

`#include <ArnServer.hpp>`

### Public Types

- enum E { NetSync }

### 10.28.1 Detailed Description

Definition at line 60 of file ArnServer.hpp.

### 10.28.2 Member Enumeration Documentation

#### 10.28.2.1 enum ArnServer::Type::E

**Enumerator:**

***NetSync***

Definition at line 61 of file ArnServer.hpp.

The documentation for this struct was generated from the following file:

- src/ArnServer.hpp (1.0.0)

## 10.29 XStringMap Class Reference

Container class with string representation.

`#include <XStringMap.hpp>`

### Public Member Functions

- XStringMap (QObject *parent=0)
- XStringMap (const QByteArray &xString, QObject *parent=0)
- ~XStringMap ()
- int size () const
- void clear ()
- int indexOf (const char *key, int from=0) const
- int indexOf (const QByteArray &key, int from=0) const
- int indexOf (const QString &key, int from=0) const
- int indexOfValue (const QByteArray &value, int from=0) const
- int indexOfValue (const QString &value, int from=0) const
- int maxEnumOf (const char *keyPrefix) const
- XStringMap & add (const char *key, const QByteArray &val)

- XStringMap & add (const char *key, const char *val)
- XStringMap & add (const char *keyPrefix, uint eNum, const QByteArray &val)
- XStringMap & add (const QByteArray &key, const QByteArray &val)
- XStringMap & add (const char *key, const QString &val)
- XStringMap & add (const char *keyPrefix, uint eNum, const QString &val)
- XStringMap & add (const QByteArray &key, const QString &val)
- XStringMap & add (const QString &key, const QString &val)
- void set (int i, const QByteArray &val)
- void set (const char *key, const QByteArray &val)
- void set (const char *key, const char *val)
- void set (const QByteArray &key, const QByteArray &val)
- void set (const char *key, const QString &val)
- void set (const QByteArray &key, const QString &val)
- void set (const QString &key, const QString &val)
- const QByteArray & keyRef (int i) const
- QByteArray key (int i, const char *def=0) const
- QByteArray key (const QByteArray &value, const char *def=0) const
- QByteArray key (const QString &value, const char *def=0) const
- QString keyString (int i, const QString &def=QString()) const
- QString keyString (const QString &value, const QString &def=QString()) const
- const QByteArray & valueRef (int i) const
- QByteArray value (int i, const char *def=0) const
- QByteArray value (const char *key, const char *def=0) const
- QByteArray value (const char *keyPrefix, uint eNum, const char *def=0) const
- QByteArray value (const QByteArray &key, const char *def=0) const
- QByteArray value (const QByteArray &key, const QByteArray &def) const
- QString valueString (int i, const QString &def=QString()) const
- QString valueString (const char *key, const QString &def=QString()) const
- QString valueString (const char *keyPrefix, uint eNum, const QString &def=QString()) const
- QString valueString (const QByteArray &key, const QString &def=QString()) const
- QString valueString (const QString &key, const QString &def=QString()) const
- void remove (int index)
- void remove (const char *key)
- void remove (const QByteArray &key)
- void remove (const QString &key)
- QByteArray toXString () const
- bool fromXString (const QByteArray &inXString, int size=-1)
- void setEmptyKeysToValue ()
- QStringList keys () const
- QStringList values () const
- void append (const char *key, const QByteArray &val)
- void append (const char *key, const char *val)
- void append (const char *keyPrefix, uint eNum, const QByteArray &val)
- void append (const QByteArray &key, const QByteArray &val)
- void append (const char *key, const QString &val)
- void append (const char *keyPrefix, uint eNum, const QString &val)
- void append (const QByteArray &key, const QString &val)
- void append (const QString &key, const QString &val)

## Static Public Member Functions

- static void stringCode (QByteArray &dst, const QByteArray &src)
- static void stringDecode (QByteArray &dst, const QByteArray &src)

### 10.29.1 Detailed Description

Container class with string representation.

This class can store data with a key like QMaps. There is a guarantied order of storing, i.e. its not sorted like QMaps.

The stored data can be ascii as well as binary. When converted to a XString, it's optimized for giving an easy readable representation.

The XString can be imported to the XStringMap. To get back stored values, XStringMap is Queried with the keys.

```
XStringMap xsm;
xsm.add("", "put");
xsm.add("id", "level");
xsm.add("val", QByteArray::number(12));
qDebug() << "XString: " << xsm.toXString();
```

This will print "XString: put id=level val=12"

Definition at line 64 of file XStringMap.hpp.

### 10.29.2 Constructor & Destructor Documentation

#### 10.29.2.1 XStringMap::XStringMap ( QObject * *parent* = 0 ) [explicit]

Definition at line 38 of file XStringMap.cpp.

#### 10.29.2.2 XStringMap::XStringMap ( const QByteArray & *xString*, QObject * *parent* = 0 ) [explicit]

Definition at line 45 of file XStringMap.cpp.

#### 10.29.2.3 XStringMap::~XStringMap ( )

Definition at line 53 of file XStringMap.cpp.

### 10.29.3 Member Function Documentation

#### 10.29.3.1 XStringMap & XStringMap::add ( const char * *key*, const QByteArray & *val* )

Definition at line 135 of file XStringMap.cpp.

#### 10.29.3.2 XStringMap & XStringMap::add ( const char * *key*, const char * *val* )

Definition at line 152 of file XStringMap.cpp.

#### 10.29.3.3 XStringMap & XStringMap::add ( const char * *keyPrefix*, uint *eNum*, const QByteArray & *val* )

Definition at line 158 of file XStringMap.cpp.

#### 10.29.3.4 XStringMap & XStringMap::add ( const QByteArray & *key*, const QByteArray & *val* )

Definition at line 168 of file XStringMap.cpp.

#### 10.29.3.5 XStringMap & XStringMap::add ( const char * *key,* const QString & *val* )

Definition at line 174 of file XStringMap.cpp.

#### 10.29.3.6 XStringMap & XStringMap::add ( const char * *keyPrefix,* uint *eNum,* const QString & *val* )

Definition at line 180 of file XStringMap.cpp.

#### 10.29.3.7 XStringMap & XStringMap::add ( const QByteArray & *key,* const QString & *val* )

Definition at line 186 of file XStringMap.cpp.

#### 10.29.3.8 XStringMap & XStringMap::add ( const QString & *key,* const QString & *val* )

Definition at line 192 of file XStringMap.cpp.

#### 10.29.3.9 void XStringMap::append ( const char * *key,* const QByteArray & *val* ) [inline]

Definition at line 127 of file XStringMap.hpp.

#### 10.29.3.10 void XStringMap::append ( const char * *key,* const char * *val* ) [inline]

Definition at line 129 of file XStringMap.hpp.

#### 10.29.3.11 void XStringMap::append ( const char * *keyPrefix,* uint *eNum,* const QByteArray & *val* ) [inline]

Definition at line 131 of file XStringMap.hpp.

#### 10.29.3.12 void XStringMap::append ( const QByteArray & *key,* const QByteArray & *val* ) [inline]

Definition at line 133 of file XStringMap.hpp.

#### 10.29.3.13 void XStringMap::append ( const char * *key,* const QString & *val* ) [inline]

Definition at line 135 of file XStringMap.hpp.

#### 10.29.3.14 void XStringMap::append ( const char * *keyPrefix,* uint *eNum,* const QString & *val* ) [inline]

Definition at line 137 of file XStringMap.hpp.

#### 10.29.3.15 void XStringMap::append ( const QByteArray & *key,* const QString & *val* ) [inline]

Definition at line 139 of file XStringMap.hpp.

#### 10.29.3.16 void XStringMap::append ( const QString & *key,* const QString & *val* ) [inline]

Definition at line 141 of file XStringMap.hpp.

#### 10.29.3.17 void XStringMap::clear ( )

Definition at line 66 of file XStringMap.cpp.

#### 10.29.3.18 bool XStringMap::fromXString ( const QByteArray & *inXString,* int *size* = −1 )

Definition at line 500 of file XStringMap.cpp.

#### 10.29.3.19 int XStringMap::indexOf ( const char * *key,* int *from* = 0 ) const

Definition at line 72 of file XStringMap.cpp.

#### 10.29.3.20 int XStringMap::indexOf ( const QByteArray & *key,* int *from* = 0 ) const

Definition at line 85 of file XStringMap.cpp.

#### 10.29.3.21 int XStringMap::indexOf ( const QString & *key,* int *from* = 0 ) const

Definition at line 96 of file XStringMap.cpp.

#### 10.29.3.22 int XStringMap::indexOfValue ( const QByteArray & *value,* int *from* = 0 ) const

Definition at line 102 of file XStringMap.cpp.

#### 10.29.3.23 int XStringMap::indexOfValue ( const QString & *value,* int *from* = 0 ) const

Definition at line 113 of file XStringMap.cpp.

#### 10.29.3.24 QByteArray XStringMap::key ( int *i,* const char * *def* = 0 ) const

Definition at line 258 of file XStringMap.cpp.

#### 10.29.3.25 QByteArray XStringMap::key ( const QByteArray & *value,* const char * *def* = 0 ) const

Definition at line 270 of file XStringMap.cpp.

#### 10.29.3.26 QByteArray XStringMap::key ( const QString & *value,* const char * *def* = 0 ) const

Definition at line 283 of file XStringMap.cpp.

#### 10.29.3.27 const QByteArray & XStringMap::keyRef ( int *i* ) const

Definition at line 249 of file XStringMap.cpp.

#### 10.29.3.28 QStringList XStringMap::keys ( ) const

Definition at line 460 of file XStringMap.cpp.

#### 10.29.3.29 QString XStringMap::keyString ( int *i,* const QString & *def* = `QString()` ) const

Definition at line 289 of file XStringMap.cpp.

#### 10.29.3.30 QString XStringMap::keyString ( const QString & *value,* const QString & *def* = `QString()` ) const

Definition at line 298 of file XStringMap.cpp.

#### 10.29.3.31 int XStringMap::maxEnumOf ( const char * *keyPrefix* ) const

Definition at line 119 of file XStringMap.cpp.

#### 10.29.3.32 void XStringMap::remove ( int *index* )

Definition at line 417 of file XStringMap.cpp.

#### 10.29.3.33 void XStringMap::remove ( const char * *key* )

Definition at line 431 of file XStringMap.cpp.

#### 10.29.3.34 void XStringMap::remove ( const QByteArray & *key* )

Definition at line 437 of file XStringMap.cpp.

#### 10.29.3.35 void XStringMap::remove ( const QString & *key* )

Definition at line 443 of file XStringMap.cpp.

#### 10.29.3.36 void XStringMap::set ( int *i,* const QByteArray & *val* )

Definition at line 198 of file XStringMap.cpp.

#### 10.29.3.37 void XStringMap::set ( const char * *key,* const QByteArray & *val* )

Definition at line 209 of file XStringMap.cpp.

#### 10.29.3.38 void XStringMap::set ( const char * *key,* const char * *val* )

Definition at line 219 of file XStringMap.cpp.

#### 10.29.3.39 void XStringMap::set ( const QByteArray & *key,* const QByteArray & *val* )

Definition at line 225 of file XStringMap.cpp.

#### 10.29.3.40 void XStringMap::set ( const char * *key,* const QString & *val* )

Definition at line 231 of file XStringMap.cpp.

#### 10.29.3.41 void XStringMap::set ( const QByteArray & *key,* const QString & *val* )

Definition at line 237 of file XStringMap.cpp.

#### 10.29.3.42 void XStringMap::set ( const QString & *key,* const QString & *val* )

Definition at line 243 of file XStringMap.cpp.

#### 10.29.3.43 void XStringMap::setEmptyKeysToValue ( )

Definition at line 449 of file XStringMap.cpp.

#### 10.29.3.44 int XStringMap::size ( ) const [inline]

Definition at line 72 of file XStringMap.hpp.

#### 10.29.3.45 void XStringMap::stringCode ( QByteArray & *dst,* const QByteArray & *src* ) [static]

Definition at line 543 of file XStringMap.cpp.

#### 10.29.3.46 void XStringMap::stringDecode ( QByteArray & *dst,* const QByteArray & *src* ) [static]

Definition at line 597 of file XStringMap.cpp.

#### 10.29.3.47 QByteArray XStringMap::toXString ( ) const

Definition at line 482 of file XStringMap.cpp.

#### 10.29.3.48 QByteArray XStringMap::value ( int *i,* const char * *def* = 0 ) const

Definition at line 314 of file XStringMap.cpp.

#### 10.29.3.49 QByteArray XStringMap::value ( const char * *key,* const char * *def* = 0 ) const

Definition at line 326 of file XStringMap.cpp.

#### 10.29.3.50 QByteArray XStringMap::value ( const char * *keyPrefix,* uint *eNum,* const char * *def* = 0 ) const

Definition at line 339 of file XStringMap.cpp.

#### 10.29.3.51 QByteArray XStringMap::value ( const QByteArray & *key,* const char * *def* = 0 ) const

Definition at line 352 of file XStringMap.cpp.

#### 10.29.3.52 QByteArray XStringMap::value ( const QByteArray & *key,* const QByteArray & *def* ) const

Definition at line 365 of file XStringMap.cpp.

#### 10.29.3.53 const QByteArray & XStringMap::valueRef ( int *i* ) const

Definition at line 305 of file XStringMap.cpp.

#### 10.29.3.54 QStringList XStringMap::values ( ) const

Definition at line 471 of file XStringMap.cpp.

#### 10.29.3.55 QString XStringMap::valueString ( int *i*, const QString & *def* = `QString()` ) const

Definition at line 375 of file XStringMap.cpp.

#### 10.29.3.56 QString XStringMap::valueString ( const char * *key*, const QString & *def* = `QString()` ) const

Definition at line 384 of file XStringMap.cpp.

#### 10.29.3.57 QString XStringMap::valueString ( const char * *keyPrefix*, uint *eNum*, const QString & *def* = `QString()` ) const

Definition at line 391 of file XStringMap.cpp.

#### 10.29.3.58 QString XStringMap::valueString ( const QByteArray & *key*, const QString & *def* = `QString()` ) const

Definition at line 403 of file XStringMap.cpp.

#### 10.29.3.59 QString XStringMap::valueString ( const QString & *key*, const QString & *def* = `QString()` ) const

Definition at line 410 of file XStringMap.cpp.

The documentation for this class was generated from the following files:

- src/XStringMap.hpp (1.0.0)
- src/XStringMap.cpp (1.0.0)

# Chapter 11

# File Documentation

## 11.1 doc/Description.md File Reference

## 11.2 doc/Install.md File Reference

## 11.3 doc/Internals.md File Reference

## 11.4 examples/Examples.txt File Reference

## 11.5 README.md File Reference

## 11.6 src/Arn.cpp File Reference

```
#include <iostream>
#include <QStringList>
#include <QVector>
#include <QDebug>
#include <QEvent>
#include <QMutex>
#include <QWaitCondition>
#include <QThreadStorage>
#include <QThread>
#include <QCoreApplication>
#include <QMetaType>
#include "Arn.hpp"
```

### Variables

- const bool gDebugThreading = 0
- const bool gDebugLinkRef = 0
- const bool gDebugRecInOut = 0
- const bool gDebugMonitor = 0

### 11.6.1 Variable Documentation

#### 11.6.1.1 const bool gDebugLinkRef = 0

Definition at line 49 of file Arn.cpp.

#### 11.6.1.2 const bool gDebugMonitor = 0

Definition at line 51 of file Arn.cpp.

#### 11.6.1.3 const bool gDebugRecInOut = 0

Definition at line 50 of file Arn.cpp.

#### 11.6.1.4 const bool gDebugThreading = 0

Definition at line 48 of file Arn.cpp.

## 11.7 src/Arn.hpp File Reference

```
#include "ArnLib.hpp"
#include "ArnLib_global.hpp"
#include "ArnError.hpp"
#include "ArnLink.hpp"
#include "ArnItem.hpp"
#include <QStringList>
#include <QVector>
#include <QObject>
#include <QMutex>
#include <QWaitCondition>
```

### Classes

- class ArnM

## 11.8 src/ArnClient.cpp File Reference

```
#include "ArnClient.hpp"
#include "ArnSync.hpp"
#include <QTcpSocket>
#include <QStringList>
#include <QTimer>
#include <QDebug>
```

## 11.9 src/ArnClient.hpp File Reference

```
#include "Arn.hpp"
#include "ArnLib_global.hpp"
#include "XStringMap.hpp"
#include <QObject>
#include <QAbstractSocket>
#include <QStringList>
```

### Classes

- class ArnClient

  *Class for connecting to an Arn Server.*

## 11.10 src/ArnDepend.cpp File Reference

```
#include "ArnDepend.hpp"
#include "Arn.hpp"
#include <QUuid>
#include <QTimer>
#include <QtAlgorithms>
#include <QDebug>
```

### Variables

- const char * ArnDependPath = "//.sys/Depend/"

### 11.10.1 Variable Documentation

#### 11.10.1.1 const char* ArnDependPath = "//.sys/Depend/"

Definition at line 40 of file ArnDepend.cpp.

## 11.11 src/ArnDepend.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnError.hpp"
#include "ArnItem.hpp"
#include <QList>
#include <QString>
#include <QObject>
```

### Classes

- class ArnDependOffer

  *Class for advertising that a service is available.*
- class ArnDepend

  *Class for setting up dependencis to needed services.*

## 11.12 src/ArnError.hpp File Reference

```
#include "MQFlags.hpp"
```

### Classes

- struct ArnError
- struct ArnError::StdCode

## 11.13 src/ArnItem.cpp File Reference

```
#include "ArnItem.hpp"
#include "Arn.hpp"
#include <QDataStream>
#include <QUuid>
#include <QTimer>
#include <QMetaObject>
#include <QDebug>
```

### Functions

- QTextStream & operator<< (QTextStream &out, const ArnItem &item)

### 11.13.1 Function Documentation

#### 11.13.1.1 QTextStream& operator<< ( QTextStream & *out,* const ArnItem & *item* )

Definition at line 986 of file ArnItem.cpp.

## 11.14 src/ArnItem.hpp File Reference

```
#include "ArnLib_global.hpp"
#include "ArnError.hpp"
#include "ArnLink.hpp"
#include "MQFlags.hpp"
#include <QTextStream>
#include <QObject>
#include <QString>
#include <QByteArray>
#include <QVariant>
#include <QAtomicInt>
```

### Classes

- class ArnItem

  *Handle for an Arn Data Object.*
- struct ArnItem::Mode

*General global mode of an Arn Data Object*

- struct ArnItem::SyncMode

  *The client session sync mode of an Arn Data Object*

### Functions

- QTextStream & operator<< (QTextStream &out, const ArnItem &item)

### 11.14.1 Function Documentation

#### 11.14.1.1 QTextStream& operator<< ( QTextStream & *out,* const ArnItem & *item* )

Definition at line 986 of file ArnItem.cpp.

## 11.15 src/ArnItemNet.cpp File Reference

```
#include <QDebug>
#include "Arn.hpp"
#include "ArnClient.hpp"
#include "ArnItemNet.hpp"
```

## 11.16 src/ArnItemNet.hpp File Reference

```
#include "ArnLib_global.hpp"
#include <QObject>
#include <QStringList>
#include "ArnItem.hpp"
```

## 11.17 src/ArnLib.hpp File Reference

### Variables

- const bool gDebugThreading
- const bool gDebugLinkRef
- const bool gDebugRecInOut
- const bool gDebugMonitor

### 11.17.1 Variable Documentation

#### 11.17.1.1 const bool gDebugLinkRef

Definition at line 49 of file Arn.cpp.

#### 11.17.1.2 const bool gDebugMonitor

Definition at line 51 of file Arn.cpp.

#### 11.17.1.3 const bool gDebugRecInOut

Definition at line 50 of file Arn.cpp.

#### 11.17.1.4 const bool gDebugThreading

Definition at line 48 of file Arn.cpp.

## 11.18 src/ArnLib_global.hpp File Reference

```
#include <QtCore/qglobal.h>
```

### Macros

- #define ARNLIBSHARED_EXPORT Q_DECL_IMPORT

### 11.18.1 Macro Definition Documentation

#### 11.18.1.1 #define ARNLIBSHARED_EXPORT Q_DECL_IMPORT

Definition at line 9 of file ArnLib_global.hpp.

## 11.19 src/ArnLink.cpp File Reference

```
#include "ArnLink.hpp"
#include <QDebug>
#include <limits>
```

## 11.20 src/ArnLink.hpp File Reference

```
#include "ArnLib.hpp"
#include "ArnLib_global.hpp"
#include "MQFlags.hpp"
#include <QObject>
#include <QString>
#include <QVariant>
#include <QAtomicInt>
#include <QMutex>
```

### Classes

- class ArnLink
- struct ArnLink::Type
- struct ArnLink::Flags
- struct ArnLink::NameF

## 11.21 src/ArnMonitor.cpp File Reference

```
#include "ArnMonitor.hpp"
#include "ArnClient.hpp"
#include "ArnItemNet.hpp"
#include <QDebug>
#include <QTime>
```

## 11.22 src/ArnMonitor.hpp File Reference

```
#include "ArnLib_global.hpp"
#include <QStringList>
#include <QObject>
```

### Classes

- class ArnMonitor

  *A client remote monitor to detect changes at server.*

## 11.23 src/ArnPersist.cpp File Reference

```
#include "ArnPersist.hpp"
#include "ArnPersistSapi.hpp"
#include "ArnDepend.hpp"
#include <QtSql/QSqlDatabase>
#include <QtSql/QSqlQuery>
#include <QtSql/QSqlError>
#include <QDir>
#include <QFile>
#include <QFileInfo>
#include <QDateTime>
#include <QRegExp>
#include <QStringList>
#include <QDebug>
#include <QMetaObject>
#include <QMetaMethod>
```

## 11.24 src/ArnPersist.hpp File Reference

```
#include "Arn.hpp"
#include "ArnLib_global.hpp"
#include <QMap>
#include <QList>
#include <QObject>
```

### Classes

- class ArnPersist

## 11.25 src/ArnPersistSapi.hpp File Reference

```
#include "ArnSapi.hpp"
```

## 11.26 src/ArnRpc.cpp File Reference

```
#include "ArnRpc.hpp"
#include <QMetaType>
#include <QMetaMethod>
#include <QRegExp>
#include <QVariant>
#include <QDebug>
```

### Macros

- #define RPC_STORAGE_NAME "_ArnRpcStorage"

### 11.26.1 Macro Definition Documentation

#### 11.26.1.1 #define RPC_STORAGE_NAME "_ArnRpcStorage"

Definition at line 40 of file ArnRpc.cpp.

## 11.27 src/ArnRpc.hpp File Reference

```
#include "Arn.hpp"
#include "XStringMap.hpp"
#include "ArnLib_global.hpp"
#include "MQFlags.hpp"
#include <QGenericArgument>
#include <QString>
#include <QByteArray>
#include <QObject>
```

### Classes

- class MQGenericArgument

  *Similar to QGenericArgument but with added argument label (parameter name)*
- class MQArgument< T >

  *Similar to QArgument but with added argument label (parameter name)*
- class ArnRpc

  *Remote Procedure Call.*
- struct ArnRpc::Mode

### Macros

- #define MQ_ARG(type, label, data) MQArgument<type >(#type, #label, data)

  *Similar to Q_ARG but with added argument label (parameter name)*

### 11.27.1 Macro Definition Documentation

#### 11.27.1.1 #define MQ_ARG( *type, label, data* ) MQArgument<type >(#type, #label, data)

Similar to Q_ARG but with added argument label (parameter name)

Definition at line 46 of file ArnRpc.hpp.

## 11.28 src/ArnSapi.cpp File Reference

```
#include "ArnSapi.hpp"
#include <QDebug>
```

## 11.29 src/ArnSapi.hpp File Reference

```
#include "ArnRpc.hpp"
#include "ArnLib_global.hpp"
#include <QString>
#include <QByteArray>
#include <QObject>
```

### Classes

- class ArnSapi

  *Service API.*

### Macros

- #define MQ_PUBLIC_ACCESS

### 11.29.1 Macro Definition Documentation

#### 11.29.1.1 #define MQ_PUBLIC_ACCESS

**Examples:**

ChatSapi.hpp.

Definition at line 45 of file ArnSapi.hpp.

## 11.30 src/ArnScript.cpp File Reference

```
#include "ArnScript.hpp"
#include "ArnDepend.hpp"
#include "ArnMonitor.hpp"
#include <QtScript>
#include <QScriptValue>
#include <QScriptEngine>
#include <QFile>
#include <QDebug>
```

## 11.31 src/ArnScript.hpp File Reference

```
#include "Arn.hpp"
#include "ArnLib_global.hpp"
#include <QObject>
#include <QScriptable>
#include <QScriptValue>
```

### Classes

- class ArnScript

## 11.32 src/ArnScriptJob.cpp File Reference

```
#include "ArnScriptJob.hpp"
#include <QScriptable>
#include <QtScript>
#include <QScriptEngine>
#include <QFileInfo>
#include <QTimer>
#include <QEvent>
#include <QDebug>
```

### Variables

- const QEvent::Type EventQuit = QEvent::Type( QEvent::User + 0)

### 11.32.1 Variable Documentation

#### 11.32.1.1 const QEvent::Type EventQuit = QEvent::Type( QEvent::User + 0)

Definition at line 43 of file ArnScriptJob.cpp.

## 11.33 src/ArnScriptJob.hpp File Reference

```
#include "ArnScript.hpp"
#include "ArnLib_global.hpp"
#include <QScriptValue>
#include <QObject>
#include <QAtomicInt>
#include <QMutex>
```

### Classes

- class ArnScriptJob
- class ArnScriptJobFactory

  *Must be thread-safe as subclassed.*
- class ArnScriptJobControl

  *Is thread-safe (except doSetupJob)*

## 11.34 src/ArnScriptJobs.cpp File Reference

```
#include "ArnScriptJobs.hpp"
#include <QDebug>
```

## 11.35 src/ArnScriptJobs.hpp File Reference

```
#include "ArnScriptJob.hpp"
#include "ArnLib_global.hpp"
#include "MQFlags.hpp"
#include <QThread>
#include <QObject>
```

### Classes

- class ArnScriptJobs
- struct ArnScriptJobs::Type
- struct **ArnScriptJobs::JobSlot**

## 11.36 src/ArnServer.cpp File Reference

```
#include <QTcpServer>
#include <QTcpSocket>
#include <QDebug>
#include "ArnError.hpp"
#include "Arn.hpp"
#include "ArnServer.hpp"
#include "ArnSync.hpp"
```

## 11.37 src/ArnServer.hpp File Reference

```
#include "ArnLib_global.hpp"
#include <QObject>
#include "Arn.hpp"
#include "MQFlags.hpp"
```

### Classes

- class ArnServer

  *Class for making an Arn Server.*
- struct ArnServer::Type

## 11.38 src/ArnSync.cpp File Reference

```
#include "ArnSync.hpp"
#include "ArnItemNet.hpp"
#include "ArnClient.hpp"
#include <QTcpSocket>
#include <QString>
#include <QStringList>
#include <QDebug>
#include <limits.h>
```

## 11.39 src/ArnSync.hpp File Reference

```
#include "ArnItemNet.hpp"
#include "ArnClient.hpp"
#include "ArnLib_global.hpp"
#include <QObject>
#include <QByteArray>
#include <QMap>
#include <QQueue>
#include <QPointer>
#include "XStringMap.hpp"
```

### Macros

- #define ARNRECNAME "rec"

### 11.39.1 Macro Definition Documentation

#### 11.39.1.1 #define ARNRECNAME "rec"

Definition at line 46 of file ArnSync.hpp.

## 11.40 src/MQFlags.hpp File Reference

```
#include <QFlags>
```

### Macros

- #define MQ_DECLARE_FLAGS(FEStruct)

  *Flags.*
- #define MQ_DECLARE_OPERATORS_FOR_FLAGS(FEStruct) Q_DECLARE_OPERATORS_FOR_FLAGS( FEStruct::F)
- #define MQ_DECLARE_ENUM(EStruct)

  *Enums.*

### 11.40.1 Macro Definition Documentation

#### 11.40.1.1 #define MQ_DECLARE_ENUM( *EStruct* )

**Value:**

```
E  e; \
    inline EStruct(E v_ = E(0)) : e( v_)  {} \
    inline static EStruct  fromInt( int v_)  {return EStruct( E( v_));} \
    inline int  toInt()  const {return e;} \
    inline operator int()  const {return e;} \
    inline bool  operator!()  const {return !e;}
```

Enums.

Definition at line 58 of file MQFlags.hpp.

#### 11.40.1.2 #define MQ_DECLARE_FLAGS( *FEStruct* )

**Value:**

```
Q_DECLARE_FLAGS(F, E) \
    F  f; \
    inline FEStruct(F v_ = F(0)) : f( v_)  {} \
    inline FEStruct(E e_) : f( e_)  {} \
    inline static E  flagIf( bool test, E e)  {return test ? e : E(0);} \
    inline bool  is(E e)  const {return f.testFlag(e);} \
    inline FEStruct&  set(E e, bool v_ = true)  {f = v_ ? (f | e) : (f & ~e);
      return *this;} \
    inline static FEStruct  fromInt( int v_)  {return FEStruct( F( v_));} \
    inline int  toInt()  const {return f;} \
    inline operator int()  const {return f;} \
    inline bool  operator!()  const {return !f;}
```

Flags.

Definition at line 40 of file MQFlags.hpp.

#### 11.40.1.3 #define MQ_DECLARE_OPERATORS_FOR_FLAGS( *FEStruct* ) Q_DECLARE_OPERATORS_FOR_FLAGS( FEStruct::F)

Definition at line 53 of file MQFlags.hpp.

## 11.41 src/XStringMap.cpp File Reference

```
#include "XStringMap.hpp"
#include <QDebug>
```

### Functions

- void XStringMapTest ()

### 11.41.1 Function Documentation

#### 11.41.1.1 void XStringMapTest ( )

Definition at line 660 of file XStringMap.cpp.

## 11.42 src/XStringMap.hpp File Reference

```
#include "ArnLib_global.hpp"
#include <QObject>
#include <QVector>
#include <QByteArray>
#include <QStringList>
```

### Classes

- class XStringMap

  *Container class with string representation.*

### Functions

- void XStringMapTest ()

### 11.42.1 Function Documentation

#### 11.42.1.1 void XStringMapTest ( )

Definition at line 660 of file XStringMap.cpp.

# Chapter 12

# Example Documentation

## 12.1 ArnDemoChat/main.cpp

Demo Chat Client

```
#include "MainWindow.hpp"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

## 12.2 ArnDemoChatServer/main.cpp

Demo Chat Server

```
#include "ServerMain.hpp"
#include <QApplication>
#include <QDebug>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv, false);

    qDebug() << "Startar Arn Chat Server ...";
    new ServerMain;

    return a.exec();
}
```

## 12.3 ChatSapi.hpp

Demo Chat Server

```
// Copyright (C) 2010-2013 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their
      own
// licenses. ArnDemoChat is independent of these licenses; however, use of
```

```
       these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef CHATSAPI_HPP
#define CHATSAPI_HPP

#include <ArnLib/ArnSapi.hpp>


class ChatSapi : public ArnSapi
{
    Q_OBJECT
public:
    explicit ChatSapi( QObject* parent = 0) : ArnSapi( parent)  {}

signals:
MQ_PUBLIC_ACCESS
    void  pv_list();
    void  pv_newMsg( QString name, QString msg);
    void  pv_infoQ();

    void  rq_updateMsg( int seq, QString name, QString msg);
    void  rq_info( QString name, QString ver);
};

#endif // CHATSAPI_HPP
```

## 12.4 MainWindow.cpp

### Demo Chat Client

```
// Copyright (C) 2010-2013 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their
       own
// licenses. ArnDemoChat is independent of these licenses; however, use of
       these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#include "MainWindow.hpp"
```

```
#include "tmp/ui_MainWindow.h"


MainWindow::MainWindow( QWidget* parent) :
    QMainWindow( parent),
    _ui( new Ui::MainWindow)
{
    _ui->setupUi( this);
    _ui->userEdit->setFocus();
    connect( _ui->lineEdit, SIGNAL(returnPressed()), this, SLOT(doSendLine()));

    _arnClient.connectToArn("localhost");
    _arnClient.setMountPoint("//");

    _arnTime.open("//Chat/Time/value");
    connect( &_arnTime, SIGNAL(changed(QString)), this, SLOT(doTimeUpdate(
      QString)));

    _commonSapi.open("//Chat/Pipes/pipeCommon");
    _commonSapi.batchConnect( QRegExp("^rq_(.+)"), this, "chat\\1");

    _soleSapi.open("//Chat/Pipes/pipe", ArnSapi::Mode::UuidAutoDestroy
      );
    _soleSapi.batchConnect( QRegExp("^rq_(.+)"), this, "chat\\1");

    _soleSapi.pv_infoQ();
    _soleSapi.pv_list();
}


MainWindow::~MainWindow()
{
    delete _ui;
}


void  MainWindow::doTimeUpdate( QString timeStr)
{
    _ui->timeEdit->setTime( QTime::fromString( timeStr));
}


void  MainWindow::doSendLine()
{
    QString  myName = _ui->userEdit->text();
    QString  line   = _ui->lineEdit->text();
    _ui->lineEdit->clear();

    _soleSapi.pv_newMsg( myName, line);
}


void  MainWindow::chatUpdateMsg( int seq, QString name, QString msg)
{
    if (seq >= _chatNameList.size()) {
        _chatNameList.resize( seq + 1);
        _chatMsgList.resize(  seq + 1);
    }
    _chatNameList[ seq] = name;
    _chatMsgList[  seq] = msg;

    QString  text;
    for (int i = 0; i < _chatNameList.size(); ++i) {
        text += _chatNameList.at(i) + ":  " + _chatMsgList.at(i) + "\n";
    }
    _ui->textEdit->setText( text);
}


void  MainWindow::chatInfo( QString name, QString ver)
{
    _ui->appNameLabel->setText( name);
    _ui->verLabel->setText( ver);
}
```

## 12.5 MainWindow.hpp

Demo Chat Client

```
// Copyright (C) 2010-2013 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
```

```
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their
       own
// licenses. ArnDemoChat is independent of these licenses; however, use of
       these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef MAINWINDOW_HPP
#define MAINWINDOW_HPP

#include "../ArnDemoChatServer/ChatSapi.hpp"
#include <ArnLib/ArnClient.hpp>
#include <ArnLib/ArnItem.hpp>
#include <QMainWindow>
#include <QVector>

namespace Ui {
class MainWindow;
}


class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void  doSendLine();
    void  doTimeUpdate( QString timeStr);

    // Chat Requester routines
    void  chatUpdateMsg( int seq, QString name, QString msg);
    void  chatInfo( QString name, QString ver);

private:
    Ui::MainWindow *_ui;
    QVector<QString>  _chatNameList;
    QVector<QString>  _chatMsgList;

    ArnClient  _arnClient;
    ChatSapi  _commonSapi;
    ChatSapi  _soleSapi;
    ArnItem  _arnTime;
};

#endif // MAINWINDOW_HPP
```

## 12.6 ServerMain.cpp

Demo Chat Server

```
// Copyright (C) 2010-2013 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their
       own
```

```
// licenses. ArnDemoChat is independent of these licenses; however, use of
       these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#include "ServerMain.hpp"
#include <ArnLib/ArnItem.hpp>
#include <QTime>
#include <QCoreApplication>
#include <QDebug>


ServerMain::ServerMain( QObject* parent) :
    QObject( parent)
{
    _timer.start(1000);
    connect( &_timer, SIGNAL(timeout()), this, SLOT(doTimeUpdate()));

    _server = new ArnServer( ArnServer::Type::NetSync
      , this);
    _server->start();

    _arnTime.open("//Chat/Time/value");


    _commonSapi = new ChatSapi( this);
    _commonSapi->open("//Chat/Pipes/pipeCommon!", ArnSapi::Mode::Provider
      );
    _commonSapi->batchConnect( QRegExp("^pv_(.+)"), this, "chat\\1");

    ArnItem*  arnPipes = new ArnItem("//Chat/Pipes/", this);
    connect( arnPipes, SIGNAL(arnItemCreated(QString)), this, SLOT(doNewSession
      (QString)));
}


void  ServerMain::doNewSession( QString path)
{
    if (!ArnM::isProviderPath( path))  return;  // Only
       provider pipe is used

    ChatSapi*  soleSapi = new ChatSapi( this);
    soleSapi->open( path, ArnSapi::Mode::Provider);
    soleSapi->batchConnect( QRegExp("^pv_(.+)"), this, "chat\\1");
    connect( soleSapi, SIGNAL(pipeClosed()), soleSapi, SLOT(deleteLater()));
}


void  ServerMain::doTimeUpdate()
{
    _arnTime = QTime::currentTime().toString();
}


void  ServerMain::chatList()
{
    ChatSapi*  sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    for (int i = 0; i < _chatNameList.size(); ++i) {
        sapi->rq_updateMsg( i, _chatNameList.at(i), _chatMsgList.at(i));
    }
}


void  ServerMain::chatNewMsg( QString name, QString msg)
{
    _chatNameList += name;
    _chatMsgList  += msg;
    int  seq = _chatNameList.size() - 1;
```

```
    _commonSapi->rq_updateMsg( seq, name, msg);
}


void  ServerMain::chatInfoQ()
{
    ChatSapi*  sapi = qobject_cast<ChatSapi*>( sender());
    Q_ASSERT(sapi);
    sapi->rq_info("Arn Chat Demo", "1.0");
}
```

## 12.7 ServerMain.hpp

Demo Chat Server

```
// Copyright (C) 2010-2013 Michael Wiklund.
// All rights reserved.
// Contact: arnlib@wiklunden.se
//
// This file is part of the ArnDemoChat - Active Registry Network Demo Chat.
// Parts of ArnDemoChat depend on Qt 4 and/or other libraries that have their
      own
// licenses. ArnDemoChat is independent of these licenses; however, use of
      these other
// libraries is subject to their respective license agreements.
//
// The MIT License (MIT)
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
// DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
// THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//
#ifndef SERVERMAIN_HPP
#define SERVERMAIN_HPP

#include "ChatSapi.hpp"
#include <ArnLib/ArnItem.hpp>
#include <ArnLib/ArnServer.hpp>
#include <QTimer>
#include <QStringList>
#include <QObject>


class ServerMain : public QObject
{
    Q_OBJECT
public:
    explicit ServerMain( QObject* parent = 0);

signals:

public slots:

private slots:
    void  doNewSession( QString path);
    void  doTimeUpdate();

    // Chat Provider routines
    void  chatList();
    void  chatNewMsg( QString name, QString msg);
    void  chatInfoQ();

private:
    QStringList  _chatNameList;
    QStringList  _chatMsgList;
    QTimer  _timer;

    ArnItem  _arnTime;
```

```
    ArnServer*  _server;
    ChatSapi*  _commonSapi;
};

#endif // SERVERMAIN_HPP
```

# Index