

Reference Manual

Generated by Doxygen 1.3.9.1

Wed Apr 4 13:40:21 2007

Contents

1	Class Index	1
1.1	Class List	1
2	Class Documentation	3
2.1	Arc::Config Class Reference	3
2.2	Arc::MCC Class Reference	4
2.3	Arc::MCC_Dummy Class Reference	6
2.4	Arc::Message Class Reference	8

Chapter 1

Class Index

1.1 Class List

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

Arc::Config (The Config (p. 3) class)	3
Arc::MCC (The base class for all Message (p. 8) Chain Components)	4
Arc::MCC_Dummy (A dummy class illustrating how to extend the MCC (p. 4) class)	6
Arc::Message (The Message (p. 8) class)	8

Chapter 2

Class Documentation

2.1 Arc::Config Class Reference

The **Config**(p. 3) class.

```
#include <MCC_Dummy.h>
```

2.1.1 Detailed Description

The **Config**(p. 3) class.

Only a dummy **Config**(p. 3) class instead of the real thing.

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

- MCC_Dummy.h

2.2 Arc::MCC Class Reference

The base class for all `Message`(p. 8) Chain Components.

```
#include <MCC_Dummy.h>
```

Public Member Functions

- `MCC ()`
A constructor.
- `virtual ~MCC ()`
The destructor.
- `virtual MCC_Status process (Message &request, Message &response)=0`
The method that processes an incoming request.

2.2.1 Detailed Description

The base class for all `Message`(p. 8) Chain Components.

All `Message`(p. 8) Chain Component (`MCC`(p. 4)) classes will extend this class. It's main purpose is to define an interface for communication between MCCs, i.e. the `process`(p. 4) method.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 Arc::MCC::MCC () [inline]

A constructor.

This is the constructor. It does nothing since there are no attributes to initialize.

2.2.2.2 virtual Arc::MCC::~~MCC () [inline, virtual]

The destructor.

This is the destructor. It does nothing since there is nothing that needs to be cleaned up.

2.2.3 Member Function Documentation

2.2.3.1 virtual MCC_Status Arc::MCC::process (Message & request, Message & response) [pure virtual]

The method that processes an incoming request.

This method is called by the preceding `MCC`(p. 4) in a chain when an incoming request needs to be processed. The advantage of sending out the response through a reference parameter is that no new `Message`(p. 8) object is created, as would be the case if the response was sent as a return value. The problem with creating new message objects is that it either involves a shallow copy of the payload (which may result in memory leaks or "dangling" pointers when the copy

is deallocated) or a deep copy of the payload (which may be an expensive operation or even impossible in case of streams).

Parameters:

request The incoming request that needs to be processed.

response A message object that will contain the response of the request when the method returns.

Returns:

An object (integer) representing the status of the call, zero if everything was ok and non-zero if an error occurred. The precise meaning of non-zero values have to be decided.

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

- MCC_Dummy.h

2.3 Arc::MCC_Dummy Class Reference

A dummy class illustrating how to extend the `MCC`(p. 4) class.

```
#include <MCC_Dummy.h>
```

Public Member Functions

- `MCC_Dummy (Config *cfg)`
A constructor.
- `~MCC_Dummy ()`
The destructor.
- `virtual MCC_Status process (Message &request, Message &response)`
The method that processes an incoming request.

2.3.1 Detailed Description

A dummy class illustrating how to extend the `MCC`(p. 4) class.

This is just a dummy `MCC`(p. 4) class that does nothing. The sole purpose of it is to illustrate how to extend the `MCC`(p. 4) class and coarsely what an implementation of the `process()`(p. 6) method could look like.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 Arc::MCC_Dummy::MCC_Dummy (Config * cfg)

A constructor.

This is the constructor. It should initialize the `next_` attribute to point to the `MCC`(p. 4) that is the successor of the present `MCC`(p. 4).

Parameters:

cfg A configuration object. No details are known yet.

2.3.2.2 Arc::MCC_Dummy::~~MCC_Dummy ()

The destructor.

This is the destructor. Should it delete (deallocate) the `MCC`(p. 4) pointed to by `next_`? If not, there may be a memory leak. If it does, there may be "dangling pointers" left in case several chains are merged to that `MCC`(p. 4).

2.3.3 Member Function Documentation

2.3.3.1 MCC_Status Arc::MCC_Dummy::process (Message & request, Message & response) [virtual]

The method that processes an incoming request.

See the corresponding method in the **MCC**(p. 4) class for a thorough description.

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

- `MCC_Dummy.h`
- `MCC_Dummy.cpp`

2.4 Arc::Message Class Reference

The **Message**(p. 8) class.

```
#include <MCC_Dummy.h>
```

2.4.1 Detailed Description

The **Message**(p. 8) class.

Only a dummy **Message**(p. 8) class instead of the real thing.

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

- MCC_Dummy.h

Index

~MCC

Arc::MCC, 4

~MCC_Dummy

Arc::MCC_Dummy, 6

Arc::Config, 3

Arc::MCC, 4

~MCC, 4

MCC, 4

process, 4

Arc::MCC_Dummy, 6

~MCC_Dummy, 6

MCC_Dummy, 6

process, 6

Arc::Message, 8

MCC

Arc::MCC, 4

MCC_Dummy

Arc::MCC_Dummy, 6

process

Arc::MCC, 4

Arc::MCC_Dummy, 6