



The framework in brief

Component-based programming framework
Written in C++, uses Qt4/5
Multiplatform: Unix, Linux, Windows
Extensible engine able to load and use at runtime:

- component libraries;
- new data types;
- exogenous component systems.

Introduces simple component and application models.
Website: <http://arcs.ibisc.univ-evry.fr>

Component model

Similar to Qt's metaobjects (<http://qt-project.org/>).
Component inputs: *slots*, outputs: *signals*
Communication: *synchronous* via *signal/slot connection*

Application model

An *application* is consisting of two parts:

- A *contextual* part;
- A *configurational* part.

A *contextual part* is composed of:

- A set of *component libraries* to load;
- A *component pool*;
- A *constant pool*.

A *configurational part* is a set of concurrent *processes*.

A *process* is controlled by a *statemachine* and is composed of a set of *operational configurations* (bound to states of statemachine) called *sheets*. A *sheet* contains:

- *pre-connection invocations* to configure components;
- *connections* to set the operational configuration;
- *post-connection* invocations to run the configuration;
- *cleanup* invocations to restore component states.

Framework parts

arcseengine: parses and runs application descriptions;
arcslibmaker: library development assistant;
arcswizard: graphical front-end to arcseengine;
arcsbuilder: builds component libraries needed by applications;
arcseditor: application graphical editor;
arcs1to2: ports applications and libraries;
libarcs.so|arcs.dll : main library;
libarcsguiw.so|arcsguiw.dll : helper library for gui mode;
ARCSDIR : environment variable needed by arcslibmaker (should indicate the path where ARCS is installed);
ARCSBUILDPATH : environment variable needed by arcsbuilder (should indicate the path where component library sources are stored).

Extending the engine

Declaring a native component

```
#include <QObject>

// QObject must be a component ancestor
class MyComponent : public QObject
{
    Q_OBJECT // mandatory
public:
    // mandatory constructor
    MyComponent(QObject* parent=0);

public slots:
    void mySlot();

signals:
    void mySignal();
};
```

Defining a component library (unix systems)

1. Prepare components source files;
2. Run arcslibmaker (produces a project);
3. Edit XML library description (.a1x file);
4. Run qmake (produces a makefile);
5. Run make to compile.

Integrating new data types

Subclass ARCSTypeFactoryTemplate<MyNewType>.

```
#include <arcs/arcslibtoolkit.h>

class ARCSTypeFactoryTemplate_MyNewType :
public ARCSTypeFactoryTemplate<MyNewType>
{
public:
    virtual QString getTypeName() const {
        // returns the type name for ARCS
    }
protected:
    virtual MyNewType parse(QString s) {
        // returns data constructed from s
    }
    virtual QString serialize(MyNewType mnt) {
        // returns a QString serializing mnt
    }
};
```

An optional step is to make this data type known by Qt as well: Q_DECLARE_METATYPE(MyNewType)

Integrating exogenous component systems

Subclass :

- ARCSAbstractFamily to register the appropriate component factories;
- ARCSAbstractComponent to define an ARCS component compatible behavior.

Supported native types

void, boolean, int, short, long, float, double, string, constant, component, size

Special component types

ARCSTypeFactoryTemplate: component logger for debugging;
composite: component made of aggregation of components;
script: scripting component using Javascript;
statemachine: process controller (transitions can be triggered by passing tokens via slot setToken(QString));

Command line

arcslibmaker

arcslibmaker [--help] [file]

arcslibmaker has two modes, one for generating ARCS library wrappers (it needs an xml file describing the library contents), the second for adding ARCS options to Qt project files.

arcsengine

arcsengine [OPTION]... [XML_FILE]...

Overriding application mode :

- b: simple loop based applications.
- e: event loop based console applications.
- g: event loop based GUI applications.
- t: threaded application.
- te: threaded event based application.

Defining options:

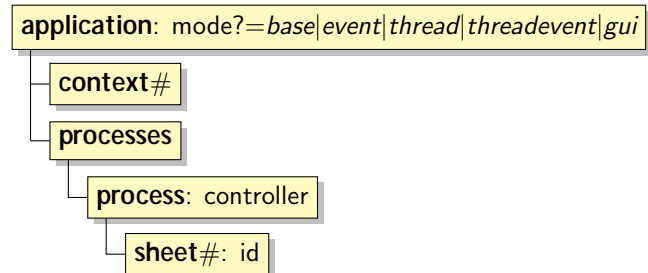
- d: define constants
- p: define a profile
- o: define a file where to dump profile

XML formats and markup hierarchy

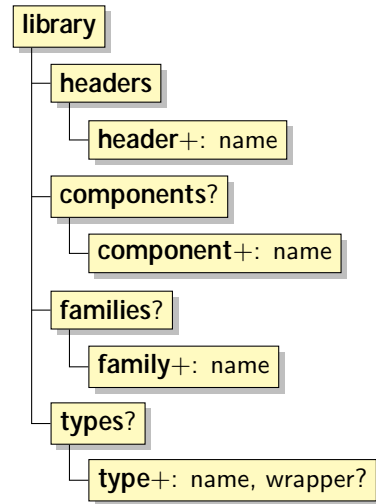
+ : at least one, ? : one or none, #: defined elsewhere.

File descriptions

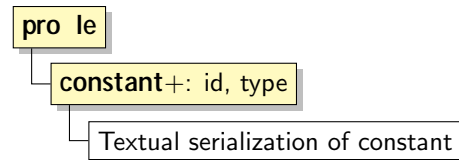
Application (file parsed by arcsengine or libarcs)



Component library (file parsed by arcslibmaker)

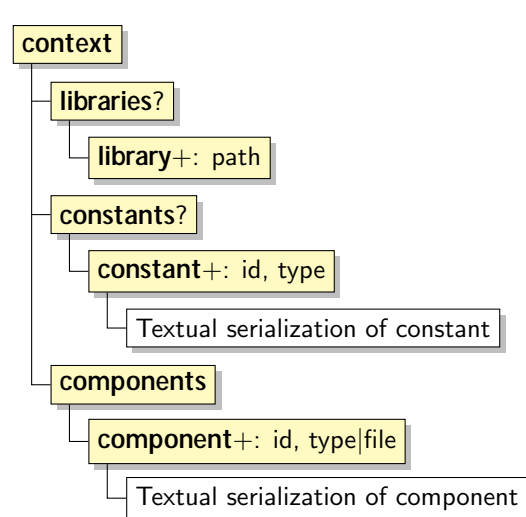


Profile (file parsed by arcsengine or libarcs)

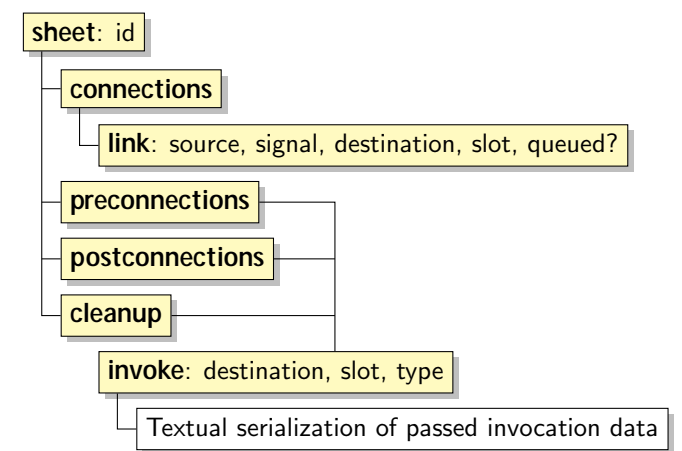


Sub-element descriptions

Context

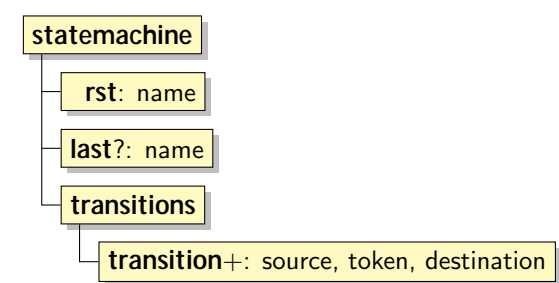


Sheet



Component descriptions

Statemachine



Composite component

