

// SC02 RPM release

// (the saga continues... complete w/real Evil Villain)

*/**

- + Is the goal something now that is precisely what was at SC02 or something useful including f90/babel-082?*
 - cca-spec-babel, dccafe now support babel 0.7.4 - 0.8.4.*
- + OS version-- do we want to add rh 7.3 to supported list?*
- + Where to host testing ?*
 - cca-forum.org user machine.*
 - cca01.csm.ornl.gov.*
 - nowhere.*
- + CVS access practicalities*
 - Send in your dsa/rsa key to revive your account on cca1.lbl.gov (man ssh-keygen)*
 - Use ssh-agent*
 - Check out clean copies of*
 - dccafe, cca-spec-babel, cca-spec-classic*
 - Now ALWAYS configure; make; make install*

+ CVS Roots are (will be):
cca1.lbl.gov:/cvs/babel/repositories
cca1.lbl.gov:/cvs/cca-spec/repository
cca1.lbl.gov:/cvs/ccaffeine/repository
cca1.lbl.gov:/cvs/cca/repositories
cca1.lbl.gov:/cvs/cca/components

*/

/*

A SIDL Parameter Port Proposal

(old wine in new, smaller bottles)

*/

```
// Leverage the CCA TypeMap and interfaces
// Simplification on the old concrete library
// Minimize existing component rewrite.
```

```
// BABEL fetching ParameterDialogService
// ( the setServices overhead )

if(svc._not_nil()) {

    tm = svc.createTypeMap();
    svc.registerUsesPort(std::string("pds"),
        std::string("caffeine.ports.ParameterDialogService"), tm);
}
```

```

// BABEL/c++ using ParameterDialogService

pds = svc.getPort("pds");
if (pds._is_nil()) {
    std::cerr << "pds not connected." << std::endl;
    return -1;
}
pds.createParameterPort(tm, pname);
::std::string title = "Test PDS for port ";
title += pname;

pds.setBatchTitle(tm, title);

pds.addRequestBoolean(tm,"noName","var to test if default group gets used",
                    "anon group",true);

pds.setGroupName(tm,"Named Set1");
pds.addRequestInt(tm,"iVar","a ranged test integer","int test", 5, 0 ,10);
pds.addRequestLong(tm,
                    "jVar","a deranged test long","long test", -50, 0 , -100);

pds.setGroupName(tm,"Named Set2");
pds.addRequestDouble(tm,
                    "dVar","a ranged test double","double test", -50, 0 , -100);
pds.addRequestFloat(tm,
                    "fVar","a ranged test float","float test", 50, -1000 , 1000);

pds.setGroupName(tm,"Named Set3");
pds.addRequestString(tm,
                    "sVar","a free test string","string any test", "some value");

pds.addRequestString(tm,
                    "sList","a choice test string","string list test", "some value");
pds.addRequestStringChoice(tm,"sList","choice 1");
pds.addRequestStringChoice(tm,"sList","choice 3");
pds.addRequestStringChoice(tm,"sList","choice 2");

pds.publishParameterPort(tm, svc);

svc.releasePort("pds");
pds = 0;

```

```

// CLASSIC fetching ParameterPortFactoryService
// ( the setServices overhead )
void TimeStamper::setServices(classic::gov::cca::Services *cc){
    svc = cc;

    // Contact the ParameterPortFactoryService to create a param shell
    classic::gov::cca::PortInfo* pinfo =
        svc->createPortInfo("cSvc", "gov.cca.ParameterPortFactoryService", 0);
    svc->registerUsesPort(pinfo);

    ConfigurableParameterFactory *cpf =
        dynamic_cast<ConfigurableParameterFactory *>(svc->getPort("cSvc"));

    if (cpf == 0) {
        return;
    }

    pp = cpf->createConfigurableParameterPort();

    // allocate params
    noName = new BoolParameter(
        "var to test if default group gets used","anon group",true);
    iVar = new IntParameter(
        "iVar","a ranged test integer","int test", 5, 0 ,10);
    jVar = new LongParameter(
        "jVar","a deranged test long","long test", -50, 0 , -100);

    dVar = new DoubleParameter(
        "dVar","a ranged test double","double test", -50, 0 , -100);
    sVar = new StringParameter(
        "sVar","a free test string","string any test", "some value");
    sList = new StringParameter(
        "sList","a choice test string","string list test", "some value");
    sList->addChoice("choice 1");
    sList->addChoice("choice 3");
    sList->addChoice("choice 2");

```

```
// CLASSIC assemble the params into the implementation shell
pp->setBatchTitle("Test PDS for port ");
pp->addRequest(noName);
pp->setGroupName("Named Set1");
pp->addRequest(iVar);
pp->addRequest(jVar);
pp->setGroupName("Named Set2");
pp->addRequest(dVar);
pp->addRequest(fVar);
pp->setGroupName("Named Set3");
pp->addRequest(sVar);
pp->addRequest(sList);

svc->releasePort("cSvc");
svc->unregisterUsesPort("cSvc");

svc->addProvidesPort(pp,
    svc->createPortInfo("configure_port", "ParameterPort", 0));
}
```

```

// CLASSIC: member data management overhead
class TimeStamper: public virtual classic::gov::cca::Component {

public:
    virtual void setServices(classic::gov::cca::Services *cc);

    TimeStamper() {
        noName = 0;
        iVar = 0;
        jVar = 0;
        dVar = 0;
        fVar = 0;
        sVar = 0;
        sList = 0;
        noName = 0;
    }

    virtual ~TimeStamper() {
        delete pp; pp = 0;
        delete noName; noName = 0;
        delete iVar; iVar = 0;
        delete jVar; jVar = 0;
        delete dVar; dVar = 0;
        delete fVar; fVar = 0;
        delete sVar; sVar = 0;
        delete sList; sList = 0;
    }

private:
    classic::gov::cca::Services* svc;
    ConfigurableParameterPort *pp;

    // compare to babel-version 1 TypeMap member.
    BoolParameter *noName;
    IntParameter *iVar;
    LongParameter *jVar;
    DoubleParameter *dVar;
    FloatParameter *fVar;
    StringParameter *sVar;
    StringParameter *sList;

};

```

```
/** This interface is exported for manipulation by other  
    components or by a user-interface (text or gui).  
    By using the port ParameterDialogService, the component  
    writer never actually has to implement this port directly.  
    */  
interface ParameterPort extends gov.cca.Port {  
/** A component may provide more than one named set of parameters.  
    */  
    array<string> getMapNames();  
  
/** Given the name, return the matching TypeMap. */  
    gov.cca.TypeMap getConfiguration(in string mapName);  
  
    void setConfiguration(in string mapName, in gov.cca.TypeMap map);  
}
```

```
interface ParameterDialogService extends gov.cca.Port
{
    /** Initialize the portData for use as a
        parameter dialog port with name portName. */
    void createParameterPort(inout gov.cca.TypeMap portData, in string portName);

    /** Define the window title for the parameter dialog. */
    void setBatchTitle(in gov.cca.TypeMap portData, in string title);

    /** Define the next tab/group title to use. */
    void setGroupName(in gov.cca.TypeMap portData, in string newGroupName);

    /** Define a parameter to the interface w/bound, dftl, etc */
    void addRequestInt(in gov.cca.TypeMap portData,
        in string name,
        in string help,
        in string prompt,
        in int default,
        in int low,
        in int high);

    void addRequestLong(...);
    void addRequestFloat(...);
    void addRequestDouble(...);
    void addRequestBoolean(...);

    void addRequestString(in gov.cca.TypeMap portData,
        in string name,
        in string help,
        in string prompt,
        in string default);

    void addRequestStringChoice(in gov.cca.TypeMap portData,
        in string name,
        in string choice);
}
```

```
/** Signal that the ParameterPort is fully defined and should  
 * now pop out on the component.  
 * The ParameterDialogService takes care of addProvidesPort  
 * and ParameterPort implementation. */  
void publishParameterPort(in gov.cca.TypeMap portData,  
                          in gov.cca.Services svc);  
  
/** Cause a previously defined parameter port to go away. */  
void unpublishParameterPort(in gov.cca.TypeMap portData,  
                             in gov.cca.Services svc);
```

```
// EVENT (get/set) listener registration functions

void setUpdater(in gov.cca.TypeMap portData,
                in ParameterGetListener powner);

void setUpdatedListener(in gov.cca.TypeMap portData,
                        in ParameterSetListener powner);

void clearRequests(in gov.cca.TypeMap portData);
} // end interface ParameterDialogService
```