Babel 0.8.4 Release

Tammy Dahlgren, Tom Epperly, and Gary Kumfert

*Center for Applied Scientific Computing*

Common Component Architecture Working Group

April 10, 2003

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

UCRL-PRES-152685
What's new in 0.8.2 (26 March 2003)

- Completed Fortran 90 phase II
  - A major improvement in look-and-feel
- Improved documentation
- Changed FORTRAN 77 cast function
- `--comment-local-only` option
Wait there's more in 0.8.4 (7 April 2003)

- Fixed F90 name mangling bug in 0.8.2
- Configuration improvements
- Doc comments for enumerated types
- C++ array binding changes
- More regression tests
Before 0.8.2, object & array references were integer*8's

Now object & array references are F90 derived types

Examples

use gov_cca_Port
use SIDL_BaseException
use gov_cca_Port_array

type(gov_cca_Port_t) :: port
type(SIDL_BaseException_t) :: excpt
type(gov_cca_Port_a) :: portArray
Generated F90 files

- Vector_fStub.c
- Vector_IOR.h
- Vector_IOR.c
- Vector_fSkel.c
- Vector.sidl
- Vector_Impl.F90
- Vector_Mod.F90
- Vector_array.F90
- Vector_fAbbrev.h
- Vector_Type.F90
- Vector.F90
Dependencies among F90 files
## F90 files & modules

<table>
<thead>
<tr>
<th>File</th>
<th>Module</th>
<th>Description</th>
<th>Edited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector_Impl.F90</td>
<td>None</td>
<td>Developer writes functions here</td>
<td>✔</td>
</tr>
<tr>
<td>Vector_Mod.F90</td>
<td>Vector_impl</td>
<td>Private data defined here</td>
<td>✔</td>
</tr>
<tr>
<td>Vector_array.F90</td>
<td>Vector_array</td>
<td>Array methods</td>
<td></td>
</tr>
<tr>
<td>Vector.F90</td>
<td>Vector</td>
<td>Object/interface methods</td>
<td></td>
</tr>
<tr>
<td>Vector_type.F90</td>
<td>Vector_type</td>
<td>Object &amp; array derived types</td>
<td></td>
</tr>
</tbody>
</table>
Impact of derived types on coding

- Distinct derived type for each class/interface enables
  - Fortran 90 overloading
    - Short method names distinguished by type
      ```fortran
      call deleteRef(obj)
      call new(obj)
      ```
  - Simple cast methods
    - Every allowable cast operation can be done in one call
      ```fortran
      call cast(port, intPort)
      ```

- Similar benefits for arrays
- Everything looks like a native F90 module
Private data pointer is a derived type

- F90 private data is a pointer to a derived type
  - Adding state data is relatively natural
  - Wrapper derived type holds pointer

```fortran
  type tutorial_Driver_private
  sequence
    ! DO-NOT-DELETE splicer.begin(tutorial.Driver.private_data)
    type(gov_cca_Services_t) :: d_services
    ! DO-NOT-DELETE splicer.end(tutorial.Driver.private_data)
  end type tutorial_Driver_private

  type tutorial_Driver_wrap
  sequence
    type/tutorial_Driver_private), pointer :: d_private_data
  end type tutorial_Driver_wrap
```
What's left to do with Fortran 90?

- Incorporate feedback from CCA & Babel users
- Use native F90 array descriptors for simple numeric types (int, long, float, double, fcomplex, dcomplex)
- Resolve name collisions with intrinsic functions
  - Example:
    size the SIDL method
    size the Fortran 90 intrinsic
  - Sun's F90 treats collisions between module functions and intrinsic functions as errors
Fortran 77 cast change

- **Old**
  
x_y_z__cast_f(obj, newtype, newobj)
  
  integer*8 obj, newobj
  character*(*) newtype

  obj was of type $x.y.z$, and it would cast it to newtype.

- **New**
  
x_y_z__cast_f(obj, newobj)
  
  integer*8 obj, newobj

  obj is any object/interface. It will be cast into type $x.y.z$ (if possible). The result is returned in newobj.

  x_y_z__cast2_f(obj, newtype, newobj)
  
  does what old _cast did.

- Similar to C and Python bindings
Miscellaneous improvements

- Reorganized and enhanced user documentation
- `--comment-local-only` for Doxygen
- Configuration improvements
  - No need for `jar -u` anymore
  - Support kaffe VM `-addclasspath`
  - Jar files stored in architecture-independent dir
  - `babel-config` script reveals configure info
- Now available in Debian unstable
Doc comments for enumerated types

- Doc comments for type and values preserved
- Added to XML representation
- Stub documentation

C Stub

```c
enum enums_car__enum {
    /**
     * A sports car.
     */
    enums_car_porsche = 911,
    /**
     * A family car.
     */
    enums_car_ford = 150,
    /**
     * A luxury car.
     */
    enums_car_mercedes = 550
};
```

SIDL

```c
// user defined values
enum car {
    /**
     * A sports car.
     */
    porsche = 911,
    /**
     * A family car.
     */
    ford = 150,
    /**
     * A luxury car.
     */
    mercedes = 550
};
```
C++ array binding change

<table>
<thead>
<tr>
<th>SIDL type</th>
<th>C++ binding ≤ 0.8.2</th>
<th>C++ binding ≥ 0.8.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>array&lt;int&gt;</td>
<td>SIDL::array&lt;int&gt;</td>
<td>SIDL::array&lt;int32_t&gt;</td>
</tr>
<tr>
<td>array&lt;long&gt;</td>
<td>SIDL::array&lt;long&gt;</td>
<td>SIDL::array&lt;int64_t&gt;</td>
</tr>
</tbody>
</table>

Similarity between array and value type was judged more important than similarity to SIDL type
Testing changes

- Added SIDL & XML backend testing
- Add F90 driver for CCA example
- Total tests: 9981
What to expect in the future

- Assertion checking in SIDL
- Fortran 90 Phase III (or incremental improvements)
- RMI/Integration