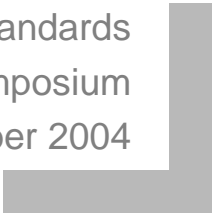


A thick, grey L-shaped bar in the top-left corner of the slide.

The 2008 COBOL Standard

Major Enhancements

Don Schricker, Director of Standards
Dutch COBOL Symposium
22 October 2004

A thick, grey L-shaped bar in the bottom-right corner of the slide.

A thick, grey L-shaped graphic in the top-left corner of the slide.


Overview

- Major Enhancements
- Schedule



A thick, grey L-shaped decorative bar in the top-left corner of the slide.

Major Enhancements for 2008


- Dynamic-capacity tables
 - Any-length elementary items
 - Function pointers
 - Increased size limit on non-numeric literals
 - Locale on upper and lower case functions
 - Structured constants
 - ISO 8601 Date/Time Support
 - XML TR
 - Collection Class TR
 - Finalizer TR, if feedback is positive
- 
- A thick, grey L-shaped decorative bar in the bottom-right corner of the slide.

Dynamic-Capacity Tables – An Example

```
1 family-record.  
3 family OCCURS DYNAMIC  
  INITIALIZED.  
5 family-name PIC X(30).  
5 childs-name PIC X(30)  
  OCCURS DYNAMIC TO 10  
  CAPACITY IS child-count.  
5 postal-code PIC X(20).
```



Dynamic-capacity Tables

- Signified by the keyword DYNAMIC
 - No FROM phrase to specify minimum, so initial size is zero.
 - New entries are automatically created when receiving operand
 - New entries are INITIALIZED, if specified
 - More than specified in the TO phrase is allowed, but it raises an exception
 - The system maintains the current size in the item specified in the optional CAPACITY phrase: child-count
 - The value of these items can be changed with a SET statement
 - Entire tables can be moved or filled
 - SORT and SEARCH work
- 




Handling string data

- COBOL allows data items of any given length, for alphanumeric or national items
- STRING and UNSTRING statements
- How many think that COBOL has excellent string handling capabilities?





Handling String Data - continued

- Have you ever tried to pass a string to another language, or receive one as an argument from another language?
 - Did it need to be terminated by a NULL character or LOW-VALUE?
 - Preceded by its length?
 - Applications today use a mix of programming languages
- 

Any-length Elementary Items – An Example

1 Any-length-examples.

```
3 alpha pic X any length.
```

```
3 nat1 pic N any length.
```

```
3 bool pic 1 any length.
```

```
3 alpha2 pic X any length
```

```
indirect.
```

```
3 bool2 pic 1 any length
```


```
limit is 64.
```

```
3 nat12 pic N any length
```

```
specified binary length to
```



Any-length Elementary Items

- Can specify maximum length, else unlimited size
 - When receiving item, the length automatically adjusted
 - When sending item, acts as a fixed-length item
 - The Intrinsic function LENGTH returns its current length
 - Length set to zero by
 - INITIALIZE
 - MOVE SPACES TO it
 - May be moved as part of a group, as long as there are corresponding variable length items in both groups
- 

A grey L-shaped graphic in the top-left corner of the slide.


Function pointers

- Similar to program-pointers in the 2002 standard
- A feature in many programming languages
- Helpful in inter-language communication – unable to make use of some Application Programming Interfaces (APIs) without them





Increased Maximum Size of Non-numeric Literals

- In the 2004 standard
 - Alphanumeric and national literals limited to 160 characters
 - Text-words in pseudo-text limited to 322 characters (COPY and REPLACE)
 - Unable to specify a table of corresponding characters even for all of the 256 ASCII characters
 - In the 2008 standard
 - Alphanumeric and national literals limited to 8191 characters
 - Text-word limited to 65535 characters
- 



Locale on upper and lower case functions

UPPER-CASE and LOWER-CASE unable to tailor mapping between
upper-case and lower-case letters

Add optional reference to a locale






Structured constants

- Some compilers can optimize code if they know that the value of a storage location will not change
- Some users want to define an initialized area and do a group move when reinitializing instead of using the INITIALIZE statement to move field-by-field
- Add CONSTANT RECORD clause to data description
- Contents of data item as if
 - INITIALIZE ... WITH FILLER ALL TO VALUE






ISO 8601 Date/Time Support

- WG4 received request for equivalent of CURRENT-DATE function in ISO date format
 - CURRENT-DATE function has both time and date components
 - These ISO formats were added:
 - Basic and extended calendar, ordinal and week date
 - Basic and extended local, UTC and offset time formats
 - Basic and extended combined date and time formats
 - Description of Standard Numeric Time Form added
 - Numeric seconds past midnight
 - Analogous to existing Integer Date Form
 - 10 new intrinsic functions
 - TEST-FORMATTED-DATETIME does not raise argument error
- 




ISO 8601 Date/Time Support

- **CURRENT-FORMATTED-DATE**
 - Analogous to CURRENT-DATE function
 - Input: date/time format
 - Output: date/time in specified format
 - **FORMATTED-DATE**
 - Analogous to DATE-OF-INTEGGER and DAY-OF-INTEGGER
 - Input: a date format and a date in integer date form
 - Output: input date formatted as specified
 - **FORMATTED-DATETIME**
 - Analogous to DATE-OF-INTEGGER and DAY-OF-INTEGGER, except time is included, as well as date
 - Input: a date/time format, date, time, optional UTC offset
 - Output: Combined date and time formatted as specified
- 




ISO 8601 Date/Time Support - continued

- **FORMATTED-DATE**
 - Analogous to DATE-OF-INTEGGER and DAY-OF-INTEGGER except for times
 - Input: time format, time, optional UTC offset
 - Output: time in specified format
 - **INTEGER-OF-FORMATTED-DATE**
 - Analogous to DATE-OF-INTEGGER and DAY-OF-INTEGGER
 - Input: a date/time format and a combined date and time value
 - Output: date in integer date form
 - **LOCALE-TIME-FROM-SECONDS**
 - Analogous to LOCALE-TIME
 - Input: a time, optionally a locale
 - Output: time according to the requirements of the locale
- 




ISO 8601 Date/Time Support - continued

- **SECONDS-FROM-FORMATTED-TIME**
 - Analogous to INTEGER-OF-DATE and INTEGER-OF-DAY except for times
 - Input: format, formatted value
 - Output: standard numeric time form
 - **SECONDS-PAST-MIDNIGHT**
 - Output: current local time of day is standard numeric time form
 - **SECONDS-FROM-FORMATTED-TIME**
 - Analogous to INTEGER-OF-DATE and INTEGER-OF-DAY
 - Input: format, formatted value
 - Output: standard numeric time form
- 




ISO 8601 Date/Time Support - continued

- **COMBINED-DATETIME**
 - Provides implementation-independent standardized timestamp
 - Input: date, time
 - Output: date + (time / 100000)
 - **TEST-FORMATTED-DATETIME**
 - Verifies consistency between date/time and format
 - Output: zero if format and data match; otherwise, position of error
- 



XML Technical Report

- XML has become the backbone of today's world of e-commerce
 - XML provides a way of sharing data between applications in a loosely-coupled environment
 - Create, read and update XML documents
 - Based on COBOL file handling syntax
 - Open, read and write XML documents in a familiar way
 - Handle dynamic XML documents
 - Map XML tags to COBOL data items
 - Process XML concepts such as namespaces and attributes
- 

Example – XML: Environment and Data Divisions

```
select xml-file assign to  
"myfile"  
  organization is xml  
  access is xml  
  document-type is external  
schema  
  "http://www.schemalocation.com  
/claim"  
  file status is filestat.
```



Example – XML: Procedure Division

procedure division.

Start-here.

Open xml-file.

initiate xml-file

Read xml-file element is x-
customer

Calculate-premium.

*> Calcule the premium

Update-XML-file.

rewrite x-customer



A thick, grey L-shaped decorative bar in the top-left corner of the slide.

Why Collections?

- Object orientation was added in the 2002 standard
- Most object-oriented languages have collections to manage sets of related object references
- Collections do not have a fixed size on the number of members





Collections

- Group of object references
- Universal (untyped) object references
- Elements accessed directly or through iterators



A large, light gray L-shaped graphic in the top-left corner of the slide.

Collection Class

- Object references retrievable in order added to collection
- Can compare itself to other collections
- Can clone itself
- Object references may be deleted
- Entire collection may be emptied



A thick, grey L-shaped decorative bar in the top-left corner of the slide.


Ordered Collection Class

- When adding object references, they may be positioned relative to other object references in the collection
- Retrievable sequentially in order they are positioned in collection





Keyed Collection Class

- When adding object references, they are associated with a national data item
 - Retrievable directly using the associated national data item
 - May be deleted using the associated national data item
- 

A thick, grey L-shaped decorative bar in the top-left corner of the slide.

Iterator Class

- More than one iterator may be associated with a collection
- Each iterator may return object references from the collection in a different sequence
- An Iterator invalid if the membership of the associated collection is changed, except through that iterator





CollectionException Class

- Raised to describe errors that occur during execution of Collection class methods
- Properties include class name and method name of the method that raised the error






ClassName Method

- Collection class Technical Report also added the ClassName method to the BASE class that is defined in the 2002 COBOL standard.
- Contents of data item as if
 - INITIALIZE ... WITH FILLER ALL TO VALUE






Finalizer Technical Report

- Finalizer Technical Report approved by ISO in 2002
 - Finalizers specify a method that gets control before an object is destroyed by the garbage collector.
 - Finalizers typically free associated resources, such as files and databases
 - If feedback on implementation of the technical report is positive, it will be included in the 2008 standard
- 

A large, light gray L-shaped graphic in the top-left corner of the slide.

Schedule for the 2008 standard

- July 2005 - Working Draft for WG4 review
 - August 2006 – First Committee Draft for SC22 review
 - October 2007 – Final Committee Draft for SC22 review
 - June 2008 – Draft International Standard for JTC1 approval
 - September 2008 – Published standard
- 
- A large, light gray L-shaped graphic in the bottom-right corner of the slide.

A thick, grey L-shaped graphic in the top-left corner of the slide.

Further Information

- External information about the COBOL standard
www.cobolstandards.com
- J4's current documents
www.cobolportal.com/J4
- Email me:
Don.Schricker@MicroFocus.com

