Base Language - Support #6854

base language tests

10/18/2022 06:47 PM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Marian Edu	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 10/20/2022 02:07 PM - Greg Shah

We need testcases that explore the following as comprehensively as possible. The intention is to provide complete coverage of the 4GL compatibility of the "base language" support.

- Expressions
  - literals of all data types and unknown value
  - all primitive data types (e.g. all BDTs)
  - operators including precedence (all except CONTAINS and :: which are database related)
    - logical OR
    - logical AND
    - bitwise OR
    - bitwise XOR
    - bitwise AND
    - logical NOT
    - bitwise NOT
    - =, EQ, <>, NE, <, LT, >, GT, ≤, LE, ≥, GE, MATCHES, BEGINS, (ignore CONTAINS)
    - binary +, binary -
    - *, /, MODULO
    - unary +, unary -
    - : (ignore ::)
    - ()
  - function calls
  - handle-based attributes and methods, including chaining (chaining with handle from a var, temp-table field, method/function call, etc)
  - system handles, "instance" handles and system handles that are referenced by instance handles
  - OO properties, data members, class event references, method calls including chaining and both instance and static references
  - extent subscripting
    - for both complex expressions (including chaining in OO vars/properties/methods) and literals
    - out of bounds subscripts (above and below)
    - ranges (e.g. var[1 for 2], to the degree they can be referenced in expressions)
- Assignments
  - = assignment operator
  - ASSIGN statement (except database-specific/UI-specific forms)
  - batching (are there any implications, perhaps on UNDO behavior?)
  - NO-ERROR
- Built-in Functions
  - all non-database, non-UI built-ins
  - include "special" ones like DYNAMIC-FUNCTION, DYNAMIC-NEW, DYNAMIC-PROPERTY, NEW, CAST, TYPE-OF, IF and others which take non-regular syntax (things that cannot normally exist in an expression)
- "Global Variables" (built-in functions which don't take parameters), including assignment for the 4 that are not read-only (CURRENT-LANGUAGE, PROMSGS, PROPATH and TERMINAL)
- Variable Definitions
  - DEFINE VARIABLE - include all permutations of definition options, with both the source and the target of a LIKE clause.
  - inline AS or LIKE clause in format phrase
  - inline AS or LIKE clause in a MESSAGE SET or MESSAGE UPDATE stmt
  - inline with UPDATE/SET/PROMPT-FOR i AS INT
- Widget Pools
  - named/unnamed widget pools
  - test all widgets via the CREATE statement, and their state after the widget pool has been deleted (implicitly or explicitly)
- Accumulators
  - accum function and statement, sub- cases

    - no UI or database access
    - there are extensive tests in testcases/uast/accum, although some use UI, the others can be re-written for automation.
- Top-Level Block Definitions
    - this can work in conjuction with the 'Control Flow' section
    - for OO blocks:
        - interfaces, abstract classes, inheritance
        - enums, flagenums
        - constructor (static or instance)
        - destructor
        - methods (static or instance)
        - method/constructor override/overload - including the dataset/dataset-handle/table/table-handle/buffer overload
        - all access modifiers
        - all return types, including extent
    - for procedure blocks:
        - function definitions (including IN SUPER, IN handle, FORWARD), all return types, including extent
        - procedure definitions (including IN SUPER)
        - all allowed access modifiers
- Control Flow
    - "inner blocks" (not FOR blocks/FOR loops, EDITING or triggers which can be handled in the database or UI tasks)
        - DO
        - REPEAT
    - "top-level" blocks
        - internal procedures
        - external procedures
        - user-defined functions
        - OO methods
        - constructors
        - destructors
        - property getters, explicit or implicit (including fixed and indeterminate extent).
        - property setters, explicit or implicit (including fixed and indeterminate extent).
    - PUBLISH, SUBSCRIBE and UNSUBSCRIBE
    - this-procedure/source/target-procedure handles
        - there are (extensive) tests in the testcases/uast/super_procs, related to super procedures - these can be adjusted to be included in CI/CD
        - this is included here as these handles are related to the control flow
    - function calls (IN SUPER, IN handle, FORWARD)
    - procedure calls (IN SUPER)
    - RUN - includes persistent and non-persistent cases
    - DYNAMIC-FUNCTION()
    - SUPER() function
    - SUPER statement (procedure)
    - SUPER class reference for method invocation
    - THIS-OBJECT class reference for method invocation
    - RUN SUPER
    - IF statement
    - CASE - including strings.
    - RETURN - including RETURN ERROR and RETURN-VALUE
    - LEAVE
    - NEXT
    - UNDO
    - PAUSE
    - STOP
    - QUIT
    - CALL handle-based resource and usage
    - block options (ON phrases, TO and WHILE clauses, TRANSACTION, STOP-AFTER; the frame phrase and DB stuff should be ignored here)
    - block properties
    - transactions/sub-transactions
    - UNDO
    - retry
    - NO-ERROR
    - conditions
    - structured error handling (try/catch/finally, OO exceptions, BLOCK-LEVEL, ROUTINE-LEVEL, interaction with legacy conditions); nested catch statements (a block with 'catch' statement from within the 'catch' or 'finally' block)
    - PROPATH() function and PROPATH statement
- Parameter Passing
    - input, input-output, output and return (where possible)
    - full range of BDT types (database features like buffers, datasets, tables... will be worked in a separate task)
    - functions
    - internal/external procs
    - OO methods
    - extents
- Type Conversion
    - various explicit cases (e.g. STRING(), DATE(), INTEGER()...) including default and non-default format strings
    - implicit conversions (as parameters, operands or assignments)

        - example: passing a CHARACTER to a INTEGER parameter at a RUN statement
    - polymorphic type support
        - dynamic invocation mechanisms that return BDT instead of a specific type (e.g. DYNAMIC-FUNCTION(), DYNAMIC-INVOKE())
- Format Strings
    - not in the UI, but in string formatting
    - cover all data types including wierd ones like OO references, handles, binary types
    - ensure cases exist for all format strings used in top 10 converted applications (can be seen in analytics reports)
- Shared Resources
    - global/new/shared options, all combinations, including error management
    - variables
    - streams
    - other resources (buffers, queries, temp-tables, frames and menus) will be handled in different tasks
- Other Extent Support
    - dynamic extents
    - range extents (outside of expressions)
    - unsubscripted extent references
    - initialization
    - asssignment
- Security
    - make sure to address all features from #3752, #3810, #4108, #4380, #6422, #6423, #6399, #6419, #6420, #6421, #6422
    - handles and system handles including all attributes and methods
        - AUDIT-CONTROL
        - AUDIT-POLICY
        - CLIENT-PRINCIPAL
        - SECURITY-POLICY
    - AUDIT-ENABLED()
    - CAN-DO()
    - CREATE CLIENT-PRINCIPAL
    - DECRYPT()
    - ENCODE()
    - ENCRYPT()
    - GENERATE-PBE-KEY()
    - GENERATE-PBE-SALT()
    - GENERATE-RANDOM-KEY
    - GENERATE-UUID
    - GET-DB-CLIENT()
    - GUID
    - HEX-DECODE()
    - HEX-ENCODE()
    - MD5-DIGEST()
    - MESSAGE-DIGEST()
    - RANDOM()
    - SET-DB-CLIENT()
    - SET-USERID()
    - SHA1-DIGEST()
    - SSL-SERVER-NAME()
    - USERID
- I18N
    - all features from our 4 phases of work (#3292, #3753, #4761, #6451) including sub-tasks and customer specifics in #4379
    - string translation support #3817 and related
    - lots of tests already exist, we need to integrate these and fill in the gaps/expand to be complete
- File System Access
    - FILE-INFO system handle
    - special "directory" stream (INPUT FROM OS-DIR)
    - OS-APPEND
    - OS-COPY
    - OS-CREATE-DIR
    - OS-DELETE
    - OS-DRIVES() function
    - OS-RENAME
    - SEARCH() function
- Streams
    - DEFINE STREAM
    - unnamed streams
    - special streams (directory, printer, terminal; ignore clipboard)
    - statements
        - INPUT FROM
        - INPUT THROUGH
        - INPUT-OUTPUT THROUGH
        - OUTPUT TO
        - OUTPUT THROUGH
        - INPUT CLOSE
        - INPUT-OUTPUT CLOSE
        - OUTPUT CLOSE
    - use the full range of access statements and functions (except redirected terminal for output/input which is in UI tests)

        - DOWN
        - EXPORT
        - IMPORT
        - PAGE
        - PAGE-NUMBER() function
        - PAGE-SIZE
        - PUT
        - SEEK statement and SEEK() function
    - all I/O options
- Shell Access
    - OS-COMMAND
    - UNIX
    - DOS
- Native Library Calls (existing tests should be complete)
- LOBS (non-database forms)
    - COPY-LOB - NUL '\0' byte needs to be tested at the begining/middle/end of the source, different codepages in CONVERT phrase.
    - LONGCHAR
    - integration with MEMPTR and RAW
- Environment and Registry/INI Access
    - LOAD, UNLOAD, USE for both Windows Registry as well as stanza-based .ini files
    - OS-GETENV
    - OS-ERROR
    - OPSYS
    - PROCESS-ARCHITECTURE
    - PROGRESS
    - PROVERSION
    - some SESSION system handle usage
- XML
    - SAX
    - DOM
- Sockets
    - server
    - client
    - SSL and non-SSL
    - various connect options

Usage of the above features in database/temp-table usage (e.g. queries, WHERE clauses, CONTAINS, :: syntax, ...) is NOT part of these tests. Instead, we will deal with those use cases in #6855.

Usage of OCX, UI/widget-based attributes/methods, COM automation and other UI features are not part of these tests. We will deal with the UI stuff in #6856.

Some of the above items (e.g. native API calls) already have testcases and can be included once they are reworked in #6858. I'm just trying to have a complete list that we can use to confirm when we are done.

I expect these tests to be split into smaller functional groupings that can be run on their own. I don't expect a single set of tests which includes all of these categories. Even larger categories like BDT tests should be split into type-specific groupings (e.g. MEMPTR and RAW tests might be a single set of tests).

We should plan for related sets of tests to be runnable which may combine features in different ways. This would require being able to run the same test from multiple test sets/targets. For example, the same built-in STRING() function would be run in BDT tests, type conversion tests, format string tests. We do need the concept of being able to run through tests avoiding duplicates (for CI/CD purposes), but the idea here is that we may need convenience groupings that allow specific testing to be targeted.

#2 - 03/14/2023 12:41 PM - Constantin Asofiei

Greg, do we need UPDATE/SET/PROMPT-FOR i AS INT definitions?

#3 - 03/14/2023 12:54 PM - Constantin Asofiei

Greg, some other notes:

- this-procedure/source/target-procedure handles - there are (extensive) tests in the old testcases project, related to super procedures - these can be adjusted to be included in CI/CD
- accumulators - do we include them here? they can work without db access.
- widget-pools - these are not specific to UI widgets, but to any resource. do we include them here?

#4 - 03/14/2023 02:28 PM - Greg Shah

> UPDATE/SET/PROMPT-FOR i AS INT

Yes, this should be added to var defs.

> this-procedure/source/target-procedure handles - there are (extensive) tests in the old testcases project, related to super procedures - these can be adjusted to be included in CI/CD

Yes

> accumulators - do we include them here? they can work without db access.

Yes

> widget-pools - these are not specific to UI widgets, but to any resource. do we include them here?

Yes

#5 - 03/14/2023 02:55 PM - Constantin Asofiei

Greg, I've edited #6854-1 - my changes are marked with 'CA:' and also 'GES:' question.

#6 - 03/14/2023 06:21 PM - Greg Shah

> do we include the dataset/table overload?

Yes, it is easier to do it here than in the database tests.

In regard to your CA: notes, I am OK with all of them. The only question I have:

> polymorphic type support (CA: e.g. passing a CHARACTER to a INTEGER parameter at a RUN statement)

This is OK, but I was thinking more about all the different DYNAMIC-FUNCTION(), DYNAMIC-INVOKE() runtime cases that return BDT and must be handled as a POLY case.

#7 - 03/15/2023 05:26 AM - Constantin Asofiei

Greg, regarding prioritizing the tests; my order would be this:

- structured error handling
- i18n
- security

#8 - 03/15/2023 05:58 AM - Greg Shah

Constantin Asofiei wrote:

> Greg, regarding prioritizing the tests; my order would be this:
>
> - structured error handling
> - i18n
> - security

I think parameter processing and extents (all forms, not just params) should also be done early on.

#9 - 03/24/2023 08:56 AM - Marian Edu

Greg, this is a rather large task, can we create sub-tasks for each topic and we refine the priority for each of those?

Other than that some of the topics are already (partially) covered by existing unit tests, most of them already converted to ABLUnit format.

#10 - 03/27/2023 03:22 AM - Greg Shah

> this is a rather large task, can we create sub-tasks for each topic and we refine the priority for each of those?

Sure. I suggest we only open sub-tasks for the next set of items to work on, rather than opening subtasks for everything and then having to manage different priorities. As we need to open new subtasks we can add new ones to open more work.

> Other than that some of the topics are already (partially) covered by existing unit tests, most of them already converted to ABLUnit format.

Anything like that would just need to be checked to:

- fill in any gaps
- meet our current standards
- make sure everything is properly automated

Some of the tests listed above might be overlapping. In those cases we don't need to duplicate the tests. We can just make sure that any common/shared set of tests can be executed from different test groupings as needed.

#11 - 06/21/2023 07:49 AM - Marian Edu

- Status changed from New to WIP