

Cobol Transformer Manual

Version 3-8-4

©1995-2000 by Siber Systems

November 2, 2000

Abstract

This manual describes *Siber Systems CobolTransformer* – an object-oriented Cobol compiler toolkit that includes:

- Cobol Parser capable of parsing the following Cobol dialects: ANSI-74, ANSI-85, OSVS, VS II, SAA, X/Open, Microsoft, Micro Focus, Ryan McFarland, RM-85, DOSVS, UNIVAC, Wang, Fujitsu, DMS DML.
- Internal Representation for Cobol programs (Program Tree) and C++ library to manipulate this representation.
- PrettyPrint library that transforms our internal representation back into beautifully indented Cobol program with fully preserved comments. PrettyPrinting is totally customizable, that is, you can directly encode your Cobol formatting rules into our table-driven PrettyPrinter.

Executive Summary. CobolTransformer lets your organization focus on particulars of your Cobol reengineering or conversion project and not worry about handling complexity of quality Cobol lexing, parsing, and generating.

To learn more about CobolTransformer, read this manual.

To get the latest CobolTransformer news, please direct your browser to <http://www.siber.com/sct/>

Copyright. © Copyright by Siber Systems 1995-2000.

No part of this software and/or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic mechanical, magnetic, optical, manual or otherwise, without the prior written permission of Siber Systems. <http://www.siber.com/>.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 8 |
| 1.1 | Manual Structure | 9 |
| 1.2 | Before We Start | 9 |
| 1.3 | System Requirements | 10 |
| 1.4 | Directory Structure | 10 |
| 2 | Program Tree | 12 |
| 2.1 | Positional and List Operation Nodes | 12 |
| 2.2 | Named Object Use and Definition Nodes | 13 |
| 2.3 | Value, Its Actual Image and Normalized Image | 13 |
| 2.4 | Memory Management for Program Tree | 13 |
| 2.5 | SctLink Smart Pointer/Link objects | 14 |
| 2.6 | Tree Operations and their Descriptors | 14 |
| 2.7 | Argument Slot | 16 |
| 2.8 | SctExpr public members | 17 |
| 2.8.1 | Basic functions | 17 |
| 2.8.2 | Image and Value | 17 |
| 2.8.3 | Operations on Arguments | 18 |
| 2.8.4 | Links | 21 |
| 2.8.5 | SourcePrint-related and positions in source | 21 |
| 2.8.6 | Tree Externalization | 22 |
| 2.8.7 | Errors, Debugging, and Monitoring | 23 |
| 2.8.8 | Public Data Members | 23 |
| 2.9 | Derived (Concrete) Node Classes | 24 |
| 2.9.1 | Constant Nodes | 24 |
| 2.9.2 | Operation Nodes | 27 |
| 2.9.3 | Named Object Use and Definition Nodes | 28 |
| 2.10 | Utility Functions | 31 |
| 2.10.1 | Null Nodes and Null Pointers | 31 |
| 2.10.2 | List Nodes | 32 |
| 2.10.3 | Object Properties | 32 |

| | | |
|----------|---|-----------|
| 2.10.4 | Picture And Usage Helpers | 32 |
| 2.10.5 | Comments Manipulation | 33 |
| 2.10.6 | Descendants and Ancestors | 33 |
| 2.10.7 | Functions to Declare and Move Data Items | 34 |
| 2.10.8 | Renaming Objects | 35 |
| 2.10.9 | Use-Definition and Symbol Table Functions | 36 |
| 2.10.10 | Miscellaneous | 38 |
| 2.10.11 | Tree Walker Functions | 38 |
| 2.11 | Length and Offset Computation | 39 |
| 2.11.1 | SctAlignInfo: Alignment information | 39 |
| 2.11.2 | Functions | 40 |
| 2.12 | SctLink: Externalizable Pointer | 41 |
| 2.12.1 | SctLink class | 41 |
| 2.12.2 | SctLinkList and SctLinkListElt classes | 42 |
| 2.13 | Miscellaneous Conversion Functions | 43 |
| 2.14 | OPTIONAL CLASSES START HERE | 43 |
| 2.15 | Expression Debug Facilities | 43 |
| 2.16 | Bag And Map Container Classes | 44 |
| 2.16.1 | template <class T> class Bag | 44 |
| 2.16.2 | template<class T, class R> class Map | 44 |
| 2.17 | Iterators | 45 |
| 2.18 | Expression Tree Manipulation Functions | 45 |
| 2.19 | Expression Classification Functions | 46 |
| 2.20 | Cobol Program Navigation Functions | 48 |
| 2.21 | Miscellaneous utility functions | 49 |
| 2.22 | Predicate and LightweightConversion classes | 50 |
| 2.23 | OPTIONAL CLASSES END HERE | 52 |
| 3 | Common Utility Classes | 53 |
| 3.1 | Class VString | 53 |
| 3.1.1 | VString class | 53 |
| 3.1.2 | Case conversions | 55 |
| 3.1.3 | Code Page conversions | 56 |
| 3.1.4 | File Name conversions | 56 |
| 3.1.5 | Miscellaneous | 56 |
| 3.1.6 | VStringList: List of Strings | 56 |
| 3.1.7 | VStringArray: Array of Strings | 57 |
| 3.1.8 | VIntArray: Array of Integers | 58 |
| 3.2 | Text Position Classes | 58 |
| 3.2.1 | Class SctTxtPosn | 58 |

| | | |
|----------|--|------------|
| 3.2.2 | Class SctTxtPosnRange | 61 |
| 3.2.3 | Class SctTxtOffs | 61 |
| 3.2.4 | Class SctTxtOffsRange | 62 |
| 3.3 | Positioned Token Classes | 62 |
| 3.3.1 | Class SctToken | 63 |
| 3.3.2 | Class SctTokenListElt | 64 |
| 3.3.3 | Class SctTokenList | 64 |
| 3.3.4 | Miscellaneous functions | 66 |
| 3.3.5 | Class SctTailedToken | 66 |
| 4 | Object Configuration | 68 |
| 4.1 | Option Descriptor | 68 |
| 4.2 | List of Option Descriptors | 70 |
| 5 | Parsing | 73 |
| 5.1 | SctParser class | 73 |
| 5.1.1 | Parser Features | 81 |
| 5.1.2 | Lexer Features | 86 |
| 6 | Cobol Code Generation | 87 |
| 6.1 | SctPrettyPrinter class | 87 |
| 6.2 | Code Generation Table | 92 |
| 6.3 | User-defined and Default tables | 93 |
| 6.4 | Formatting Actions | 93 |
| 6.5 | SourcePrint Overview | 95 |
| 6.6 | How to Use SourcePrint | 96 |
| 7 | Query Language | 98 |
| 7.1 | Query Language Reference | 98 |
| 7.2 | Query Language Examples | 110 |
| 7.3 | Query API: SctQuery | 112 |
| 8 | Transformation | 114 |
| 8.1 | Transformation ID | 114 |
| 8.2 | Transformation storage | 116 |
| 8.3 | Transformation | 118 |
| 8.3.1 | Constructors, destructors, creators, memory management | 118 |
| 8.3.2 | Transformation class properties and description | 119 |
| 8.3.3 | Init, Config, Find and Apply operations | 120 |
| 8.3.4 | References | 122 |
| 8.3.5 | Save/Load | 123 |
| 8.3.6 | Diagnostics and Debugging | 123 |

| | | |
|-----------|--|------------|
| 8.3.7 | Data members | 124 |
| 8.4 | Macros used in transformations | 125 |
| 8.4.1 | Declaration Macros | 125 |
| 8.4.2 | Transformation Body Definition | 126 |
| 8.4.3 | Macros used in Transformation Body | 126 |
| 9 | Transformation Library | 128 |
| 9.1 | Cobol Beautifier Transformation | 128 |
| 9.2 | Renumbering Transformations | 128 |
| 9.3 | Cobol Grep Transformation | 129 |
| 9.4 | File and Data Descriptor Extractors | 130 |
| 9.5 | IBM printer control characters simulation | 130 |
| 9.6 | IBM Cobol To Fujitsu Cobol Transformations | 130 |
| 9.7 | Micro Focus to Fujitsu Transformations | 131 |
| 9.8 | Loaded Transformation Runner | 136 |
| 9.9 | Finder Transformations | 136 |
| 9.10 | IBM Cobol Dialects Normalization Transformations | 136 |
| 9.11 | Visual Macro | 138 |
| 9.12 | SQL Transformations | 138 |
| 9.13 | Standard Transformations | 139 |
| 9.14 | Utility Transformations | 142 |
| 9.15 | Year 2000 Fix Transformations | 143 |
| 10 | Cobol Program Tree: Reference | 144 |
| 10.1 | Generic Operations | 146 |
| 10.2 | Cobol File | 146 |
| 10.3 | IDENTIFICATION Division | 147 |
| 10.4 | ENVIRONMENT Division | 147 |
| 10.4.1 | CONFIGURATION Section | 147 |
| 10.4.2 | INPUT-OUTPUT Section | 157 |
| 10.5 | DATA Division | 165 |
| 10.5.1 | FILE Section | 165 |
| 10.5.2 | Data Storage Sections | 169 |
| 10.5.3 | COMMUNICATION Section | 178 |
| 10.5.4 | REPORT Section | 180 |
| 10.5.5 | SCREEN Section | 184 |
| 10.6 | PROCEDURE Division | 189 |
| 10.6.1 | General Division Structure | 190 |
| 10.6.2 | Copy Statements | 193 |
| 10.6.3 | ACCEPT statement | 194 |
| 10.6.4 | ADD statement | 199 |

| | | |
|---------|------------------------------------|-----|
| 10.6.5 | ALTER statement | 200 |
| 10.6.6 | CALL statement | 200 |
| 10.6.7 | CANCEL statement | 202 |
| 10.6.8 | CHAIN statement | 202 |
| 10.6.9 | CLOSE statement | 202 |
| 10.6.10 | COMMIT statement | 203 |
| 10.6.11 | COMPUTE statement | 203 |
| 10.6.12 | CONTINUE statement | 203 |
| 10.6.13 | DELETE statement | 203 |
| 10.6.14 | DELETE FILE statement | 204 |
| 10.6.15 | DISABLE statement | 204 |
| 10.6.16 | DISPLAY statement | 204 |
| 10.6.17 | DIVIDE statement | 208 |
| 10.6.18 | ENABLE statement | 209 |
| 10.6.19 | ENTRY statement | 209 |
| 10.6.20 | EVALUATE statement | 209 |
| 10.6.21 | EXAMINE statement | 210 |
| 10.6.22 | EXEC statement | 211 |
| 10.6.23 | EXHIBIT statement | 211 |
| 10.6.24 | EXIT statement | 212 |
| 10.6.25 | EXIT PROGRAM statement | 212 |
| 10.6.26 | EXIT PERFORM statement | 212 |
| 10.6.27 | EXIT PARAGRAPH statement | 212 |
| 10.6.28 | GENERATE statement | 212 |
| 10.6.29 | GOBACK statement | 213 |
| 10.6.30 | GOTO statement | 213 |
| 10.6.31 | IF statement | 213 |
| 10.6.32 | INITIALIZE statement | 213 |
| 10.6.33 | INITIATE statement | 214 |
| 10.6.34 | INSPECT statement | 214 |
| 10.6.35 | MERGE statement | 217 |
| 10.6.36 | MOVE statement | 217 |
| 10.6.37 | MULTIPLY statement | 217 |
| 10.6.38 | NEXT-SENTENCE statement | 218 |
| 10.6.39 | NOTE statement | 218 |
| 10.6.40 | OPEN statement | 218 |
| 10.6.41 | PERFORM statement | 219 |
| 10.6.42 | PURGE statement | 220 |
| 10.6.43 | READ statement | 220 |
| 10.6.44 | READY TRACE statement | 221 |

| | | |
|---------|---------------------------------------|-----|
| 10.6.45 | RECEIVE statement | 221 |
| 10.6.46 | RELEASE statement | 222 |
| 10.6.47 | RESET-TRACE statement | 222 |
| 10.6.48 | RETURN statement | 222 |
| 10.6.49 | REWRITE statement | 222 |
| 10.6.50 | ROLLBACK statement | 223 |
| 10.6.51 | SEARCH statement | 223 |
| 10.6.52 | SEND statement | 224 |
| 10.6.53 | SERVICE RELOAD statement | 224 |
| 10.6.54 | SET statement | 224 |
| 10.6.55 | SORT statement | 226 |
| 10.6.56 | START statement | 226 |
| 10.6.57 | STOP statement | 227 |
| 10.6.58 | STRING statement | 228 |
| 10.6.59 | SUBTRACT statement | 228 |
| 10.6.60 | SUPPRESS PRINTING statement | 229 |
| 10.6.61 | TERMINATE statement | 229 |
| 10.6.62 | TRANSFORM statement | 229 |
| 10.6.63 | UNLOCK statement | 229 |
| 10.6.64 | UNDELETE statement | 229 |
| 10.6.65 | UNSTRING statement | 230 |
| 10.6.66 | USE statement | 231 |
| 10.6.67 | WRITE statement | 232 |
| 10.7 | Common Expression Subtrees | 233 |
| 10.7.1 | Leaf Operations | 234 |
| 10.7.2 | Intrinsic Function | 235 |
| 10.7.3 | Reference Modification | 235 |
| 10.7.4 | Table Element | 235 |
| 10.7.5 | Arithmetic Expression | 236 |
| 10.7.6 | Condition | 237 |
| 10.8 | Wang Extension | 241 |
| 10.8.1 | Figurative Constants | 241 |
| 10.8.2 | File Control Entry | 241 |
| 10.8.3 | File Description Entry | 242 |
| 10.8.4 | Record Description Entry | 243 |
| 10.8.5 | Statements | 246 |
| 10.9 | SQL Extension | 251 |
| 10.9.1 | Top Level Statements | 251 |
| 10.9.2 | Declarative Statements | 252 |
| 10.9.3 | WHENEVER Statement | 252 |

| | | |
|---------|--|-----|
| 10.9.4 | Connect and Transaction Statements | 255 |
| 10.9.5 | SELECT Statement | 257 |
| 10.9.6 | Data Modification Statements | 259 |
| 10.9.7 | Cursor Statements | 259 |
| 10.9.8 | Dynamic SQL Statements | 263 |
| 10.9.9 | Expression | 266 |
| 10.9.10 | Condition | 269 |
| 10.9.11 | Table Expression (Subquery) | 272 |
| 10.10 | CICS Extension | 275 |
| 10.10.1 | Top Level Statements | 275 |
| 10.10.2 | Options | 291 |
| 10.10.3 | Arguments | 302 |

Chapter 1

Introduction

When starting a Cobol reengineering, modernization or conversion project, such as Year 2000 project, an organization has to solve many technical and managerial problems.

Transformation Rules. Among technical problems, the hard one is to determine a formal set of program transformation rules. Rules like: “For every data item description in which **USAGE** is **COMP-5** or **BINARY** and **PICTURE** is **X(n)** change **PICTURE** to **9(2*n)**”.

These rules may be simple or they may be complicated, you can establish the rules by yourself, or you can hire Cobol reengineering company to help you determine what the program transformation rules are.

Manual Conversion – Just Say No. But once you have determined the transformation rules, you need to decide how to execute them and thus make conversion happen. Do you want to hire 100 Cobol programmers and make them apply the rules manually?

Many reengineering specialists today believe that performing the conversion job manually is not the way to go:

- Manual labor is prone to errors, even when executing a set of well-defined rules.
- Automation of repetitive tasks is a way of increasing productivity and decreasing costs.
- Automatic conversion can be repeated as often as you want. So if set of rules was incomplete, you can adjust the rules and fire up conversion tool one more time. Firing up 100 Cobol programmers one more time is a much costlier procedure.
- Modern and ancient Cobol programs are huge – millions lines of code. Oftentimes you just will not have enough time and money to perform transformation manually.

Automated Conversion and its Infrastructure. So it is likely that you decide to do an *automated conversion*. That is, you would hire somebody to write a *conversion application* — a program that automatically applies your conversion rules to the code.

As many IT managers learned, conversion program needs to have a number of important components that are not readily available to them:

- First, conversion program needs to “understand” Cobol code. Compiler writers call this process *parsing*. Parsing Cobol is **hard**, but it is a fairly standard procedure that is relatively

well defined in Cobol standards.

- Then you need a mechanism for expressing your conversion rules, and an infrastructure of some kind to implement these rules. Things like program representation, program tree browsing, tree modification, tree flattening (marshaling) and unflattening.
- After the program was parsed and transformation rules applied, the program has to be written out as a human-readable and maintainable Cobol source. Therefore, you need a good Cobol Code Generator too. You want to have an aesthetically pleasing program on the output of your generator, and generating aesthetically pleasing programs automatically is far from trivial.

Where to Get Conversion Infrastructure? Now you have several choices. You may decide to develop conversion infrastructure in-house. But do you have time and money to do this?

Writing a decent Cobol lexer and parser takes about 2 or 3 man-years. And this is time spent by expensive programmers that are highly skilled in compilers.

Apparently, it is much easier to use off-the-shelf solution, and Siber Systems **CobolTransformer** is the solution that you need.

With CobolTransformer you get a well-tested and maintained parser that understands all major Cobol dialects in existence today, you get a library that lets you manipulate the trees produced by the parser, and you get a Cobol PrettyPrint Code generator that is 100% customizable.

1.1 Manual Structure

We start with defining CobolTransformer internal representation for the Cobol program. This representation is based on fairly standard *expression trees* that are used today in most compilers. Tree construction and transformation is explained.

Then we proceed to the parsing process and explain how you can invoke CobolTransformer parser function from your program.

And finally we explain Cobol PrettyPrint Code Generator and show you how the Code Generation Table can be customized.

1.2 Before We Start

CobolTransformer is a powerful toolkit. As with any power tool, you need to have certain skills to use it.

Specifically, you should understand how *compilers* and *parsers* work, what *expression tree* is, and what are the standard ways of working with the trees. If you feel a bit unfamiliar with this subject, the good source would be a classic book [ASU86].

Also you should know C++, since this is the language used in accessing CobolTransformer library.

And, yes, you have to know *Cobol*. We used [Mic93], [Ame85] and [Int] as a Cobol reference, and we recommend that you read these books too.

1.3 System Requirements

Since CobolTransformer is distributed to you in source, system requirements depend a lot on what kind of computer and compiler you use.

System requirements also may be affected by the size of your application that uses the CobolTransformer library.

Our experience with **Cbl-Beau** shows that 4 Mb of *user* memory on a 486-compatible processor should be enough for most medium-size programs. On Linux it translates into 8-16 Mb physical memory on 486. On Windows NT it would be equivalent to at least 24-32 Mb memory on Pentium. The biggest Cobol programs that we encountered can take up to 25Mb of user memory which is still within limits of modern PCs.

1.4 Directory Structure

There are several directories in the CobolTransformer distribution:

- * **sct**
The CobolTransformer library itself resides here.
- * **lbase**
The base services library of CobolTransformer.
- * **trabslib**
The transformation library of CobolTransformer.
- * **smp-ibm2fsc**
Prototype of IBM OSVS/VSII Cobol to Fujitsu Cobol converter. This is not a complete tool, but rather a prototype that demonstrates use of CobolTransformer in converter of a moderate complexity.
- * **smp-gen-prog**
This demo program builds a new Program Tree and then PrettyPrints it into a file. The program demonstrates Program Tree building techniques.

The source files that are included in the **sct** and **lbase** directories of CobolTransformer distribution are listed in Figure 1.1.

String used in the “Part of SCT” column mean the following:

- * **Y**
Yes – a supported part of CobolTransformer.
- * **Extra**
Extension of the base CobolTransformer, paid for separately from the base SCT. Supported.
- * **DefExt**
Default Extension. This is not a part of CobolTransformer, just a default implementation of certain necessary functions not related to parsing or Cobol. User is encouraged to provide her own implementation of these functions.

| Source File | Header File | Part of SCT | Explanation |
|-----------------|-------------------|-------------|--|
| common.h | | Y | Common Types and Constants |
| debug.h | debug.cxx | DefExt | Debugging module |
| error.h | | Y | Error Codes |
| exiter.h | exiter.cxx | Y | SctExit function |
| file-util.h | file-util.cxx | Y | File Utilities (OS dependent) |
| machine.h | machine.cxx | Y | Machine/OS-dependent functions |
| setflags.h | setflags.cxx | Y | Command-Line and Visual Options |
| sighandler.h | sighandler.cxx | DefExt | Signal Handler |
| strtable.h | strtable.cxx | Y | String Table |
| tic.h | tic.cxx | Y | Intermediate Code (TIC) functions |
| vstring.h | vstring.cxx | Y | String Class |
| *.txt | | Text | README files |
| cics.tdl | cics.y | Extra | CICS Extension |
| clauses.h | | Y | List of Clauses |
| cobol.tdl | cobol.y | Y | Cobol Program Tree and Grammar |
| counter.h | counter.cxx | Y | Count things in Cobol Program |
| diag.h | diag.cxx | DefExt | Diagnostics module |
| dialects.h | dialects.cxx | Y | Count things in Cobol Program |
| dms-dml.tdl | dms-dml.y | Extra | DMS DML Extension |
| expr.h | expr.cxx | Y | Program Tree Library |
| expr-util.h | expr-util.cxx | Y | Expression Utilities |
| keywords.h | keywords.cxx | Y | Reserved Words |
| link.h | link.cxx | Y | InterNode Link Facility |
| makefile | | Y | Makefile for SCT library |
| opers.h | | Y | List of Operations |
| parser.h | parser.cxx | Y | Parser Interface |
| parser-excl.h | parser-excl.cxx | Y | Parser Executable Client |
| parser.exe | | Y | Parser Executable (Server) |
| pdiag.h | pdiag.cxx | DefExt | Diagnostics with Position |
| postoken.h | postoken.cxx | Y | Token That Knows Its Position |
| pprinter.h | pprinter.cxx | Y | PrettyPrinter Interface |
| pprinter-excl.h | pprinter-excl.cxx | Y | PrettyPrinter Executable Client |
| pprinter.exe | | Y | PrettyPrinter Executable (Server) |
| query.h | query.cxx | Y | Query |
| sct-xcl.lib | | Y | CobolTransformer Client Library (compiled) |
| sct-manual.* | | Y | This Manual in different formats |
| sql.tdl | sql.y | Extra | SQL Extension |
| symtab.h | symtab.cxx | Y | Symbol Table |
| transform.h | transform.cxx | Y | Transformations |
| tree.h | tree.cxx | Y | Tree, used in searches |
| tokens.h | | Y | List of Tokens |
| txtposn.h | txtposn.cxx | Y | Position in Text file |
| wang.tdl | wang.y | Extra | Wang Extension |

Figure 1.1: CobolTransformer Packing List

Chapter 2

Program Tree

In CobolTransformer we represent Cobol Program as a *Program Tree*.

Sometimes Program Tree is also called *Abstract Syntax Tree*. We do not like to use this term, because our tree reflects not so much Cobol syntax, but to a considerable degree, semantics of Cobol program.

However we preserve all syntactic elements (comments, value images) that are important for Cobol program reengineering purposes.

Program Tree Node is declared in file `expr.h` as a polymorphic C++ class `SctExpr`.

`SctExpr` nodes may not be constructed directly, because `SctExpr` is just a base class for concrete (derived) types of nodes. The concrete nodes may be constructed, each of them has a number of constructors. Program Tree consists of the concrete nodes.

Each node in the Program Tree has a list of children. There can be zero, one, or more children at every node. Nodes that do not have children are called *leaves*.

All the concrete nodes can be classified into 3 classes:

- Leaf Constant Nodes These nodes represent integer, fixed and float numeric literals, non-numeric literals, and character-strings.
- Operation Nodes These nodes represent Cobol operations. The whole Cobol program hierarchy is represented in our Program Tree using a hierarchy of such operations.
- Naming-related Nodes These nodes handle user-defined names, named objects, everything related to naming in Cobol.

2.1 Positional and List Operation Nodes

Operation nodes can be either **Positional Nodes** or **List Nodes**.

In Positional nodes every child of a node has a certain meaning that is defined by operation of this node and by relative position of the child in this node.

For instance, child number 4 (number `PROG_PROC_DIV`) of node with operation `COBOL_PROGRAM` contains Procedure Division of the Cobol Program.

In List Nodes all children are elements of a *uniform* list and children's position in the parent node does not really matter. It only matters in the sense that children follow each other in a certain order, but absolute numeric position of a child is not important.

2.2 Named Object Use and Definition Nodes

These derived nodes were introduced in Version 2.1 of CobolTransformer to implement Symbol Table and Use-Definition links.

Symbol Table is a linear list of names that is used to look up the name. Every Symbol Table element has a list of Definition nodes for this name defined by it to make look up process fast.

Named Object is a Cobol object that has a name. It can be one of: cd-name, class-name, condition-name, data-name, record-name, split-key-name, file-name, index-name, mnemonic-name, paragraph-name, section-name, program-name, report-name, routine-name, screen-name.

Every Named Object has exactly one definition represented by **SctNamedObjDef** node. All uses of the Named Object are represented by **SctNamedObjUse** nodes, and every **SctNamedObjUse** node has a link to the defining **SctNamedObjDef** node.

Since a Named Object may be referred to by Qualified Name, we also have a **SctNameUse** node that represents a single member name of Qualified Name. **SctNameUse** has a reference to Symbol Table element that represent this member name. Every **SctNamedObjUse** node has a list **SctNameUse** nodes attached to it that represent Qualified Name describing the Use exactly as it appears in the source Cobol program.

2.3 Value, Its Actual Image and Normalized Image

We will use words Image and Value fairly often in the text of this manual.

All leaf nodes in the tree have *Value*. For instance, **SctStrConst** node's value is string stored in this node, and **SctIntNumConst** node's value is integer stored in the node.

Every Value can have several *Images*. For instance, integer Value 10 may have the following images: 0010, +10, H'0A', and so on. All these images represent the same value.

Every Value has a *Normalized Image* – a string that is usually used to represent this Value. For the above example Normalized Image is 10.

Since CobolTransformer client may want to preserve exact Image of a Value and not substitute it with Normalized Image, we store the original Image of the Value in the expression node — exactly as it appears in the source Cobol program.

2.4 Memory Management for Program Tree

All nodes in all CobolTransformer expressions are dynamically allocated with the **new** operator.

Expression (or subexpression) is represented by pointer to its root node. This is why all functions that deal with expressions take **SctExpr *** as their arguments and return it as a result.

All CobolTransformer functions can be classified into 3 classes based on their role in expression memory management:

- Consumer-Producer

This function takes one or more expressions, totally consumes them, and returns the resulting expression.

Often the returned expression is constructed from the nodes that belonged to the consumed argument expressions.

If argument expressions are not reused in the resulting expression, they (or their unused parts) must be **deleted** by the consumer, or memory leak would result.

- **Modifier**

This function is given an expression as an argument. It modifies it, and it does not return anything.

Essentially this is a Consumer-Producer function in which argument and result is the same expression.

- **Browser**

This function does not modify an expression given to it, it just collects a certain data from the expression.

2.5 SctLink Smart Pointer/Link objects

CobolTransformer can produce “flattened” (or “marshaled” or “externalized”) version of the Program Tree that is used: (1) to write the Tree to a file, and/or (2) to send the Tree across the network.

The flattened form is strictly platform-independent and it can be freely transferred from one platform to another – no questions about platform endianness asked.

Since Symbol Table-related links make the Program Tree look more like a general *graph*, we cannot use regular pointers for Use-Def links – it would not be easy to externalize the regular pointers.

So the links in the tree that are not links between Parent and Child are represented using **SctLink** class. SctLink has both a memory pointer to the linked-to node and Integer Index of the linked-to node.

Every node in Program Tree has an Integer Index too, so SctLinks are written out as Integer Indices when tree is externalized. For in-memory browsing we use pointer component of SctLink, and when link needs to be flattened, the Integer Index component of SctLink goes to external file.

2.6 Tree Operations and their Descriptors

Tree operation is declared as **enum SctOper**. The list of operations in this enum is generated from file `cobol.tbl` that describes all Cobol Program Tree operations.

Class **SctOperDescr** describes the Program Tree operation. The properties stored in this class are constant properties of a given operation that do not depend on where and how operation is instantiated.

SctOperDescr has the following members:

- SctOperDescr(SctOper i_oper, char *i_word, char *i_descr, int i_is_list, int i_is_stmt, int i_arg_no, int i_fixed_arg_no)

Constructor.

- SctOper oper

Operation code of the described operation.

- `char *word`
Official Operation name.
- `char *descr`
Short Operation Description in English.
- `char *gen`
String that represents this operation in the source program. Do not use this string for pretty-printer, it keeps separate tables.
- `int is_list`
Flag: True if this a list operation, False if this is a positional operation.
- `int is_stmt`
Flag: is this a statement. Statement is a Cobol term that means unit of work in Procedure Division. We use statements only to find point in the tree where another statement can be inserted or for accounting purposes.
- `int arg_no`
Official number of arguments (children) for this operation. If this is a list operation, `arg_no` is 0, but list operation can have any number of arguments.
- `int fixed_arg_no`
Number `NF` such that for the first `NF` arguments of this operation the lexicographical order of the arguments must coincide with the order of arguments indices. If `NF` is `-1`, then all arguments must appear in order.

The table of operations is defined as `SctOperDescr SctOperDescrs[]`, but you do not need to use it directly, because we have a number of functions that provide access to the properties of a given operation.

The table of clauses is defines as `SctOpIdxName SctClauseNames[]`, but you do not need to use it directly, because we have a number of functions that provide access to the properties of a given clause.

Functions that use operation descriptors:

- `char *SctOperStr(SctOper op)`
Return official name of the operation `op`. The name is returned as a pointer to static string.
- `char *SctGenOperStr(SctOper op)`
Return code generation template for operation `op`. Sometimes this template can be used as a string representation for operation. The name is returned as a pointer to static string.
- `SctOperPutTic(SctOper op, ostream &ost)`
Write operation `op` string to ostream `ost`.
- `SctErrCode SctOperGetTic(istream &ist, SctOper &op, VString &msg)`
Read operation `op` from istream `ist`.

- `int SctIsListOper(SctOper op)`
Return True if expression operation `op` is a list expression operation (variable number of children of the same kind). Return FALSE if operation `op` is positional expression operation (fixed number of children of different kinds).
- `int SctIsStmtOper(SctOper op)`
Return True if operation `op` heads the statement. Otherwise return FALSE.
- `int SctOperArgNo(SctOper op)`
Return official number of children for a given expression operation. Return -1 if `op` is not found.
- `VString SctClauseName(SctOper op, int ix, int *not_found = NULL, const char *fmt = NULL)`
Return clause name for a clause number `ix` in positional operation `op`. Set `*not_found` to False.
Set `*not_found` to True if given clause has no name and return substitute name. You can specify your own substitute name format in string `fmt`.
- `int SctIsClauseHidden(SctOper op, int ix)`
Return True if clause number `ix` in positional operation `op` is hidden. Otherwise return False.
- `int SctOperLocate(const char *name, int *t)`
If given name is Operation Name, return Operation code for this name. If given name is Clause Name, return Clause Index for this name. If given name is not any of the above, return -1. Write type of the name to `*t`. It can be `T_IS_OPER` or `T_IS_POS`.

2.7 Argument Slot

Parent node has a number of argument slots, one for each argument (child).

The slot is used to store information that: (1) does not belong solely to the parent node, (2) does not belong solely to the child, but rather (3) belongs to a particular combination of parent and child. Argument slots were introduced to store data for `SourcePrint`.

Please do not use `SctArgSlot` constructors, since argument slots are always constructed automatically when you attach child node to the parent node. In general, argument slots are not really visible to the end user.

The argument slot has the following members:

- * `SctTokenList precs`
Source Tokens that precede this argument.
- * `SctExpr *arg`
The argument in this slot.
- * `int16 from_src`
If True, this slot originates from the program source. If False, this slot is newly generated and has no connection to source.
- * `int16 lex_ix`
Lexicographical index of this slot in the parent operation.

2.8 SctExpr public members

These are the functions used in creation and manipulation of the Program Tree.

2.8.1 Basic functions

- `SctExpr *Copy()` const
Perform a deep copy of the tree that starting at this node and return the copy of this tree.
- `~SctExpr()`
Destruct this tree.
- `void TakeOut()`
Take this sub-tree out from the big tree, severing all links with this node, but do not delete this node.
- `static void *operator new(size_t s)`
Custom `delete` that writes 0xAA into memory to make debugging of memory allocation errors easier.
- `SctOper Oper()` const
Return operation of this tree node.
- `void SetOper(SctOper oper, int reset_from_src=TRUE, int grow_shrink=FALSE)`
Change operation at this node to operation `oper`.
`SetOper()` can be called only from `SctExprOper` nodes. To convert `SctExprOper`, say, into `SctStrConst` you need to delete `SctExprOper` first and then create and link in the new `SctStrConst` node.
If `reset_from_src` is True (default), then the whole subtree starting at this node is disconnected from the source.
If `grow_shrink` is True (default is False), then the number of arguments of the new operation can be different from the number of arguments of the old operation. Otherwise the Severe Error is reported.
- `int IsNull()`
Return True, if this node is `SctNullNode`.

2.8.2 Image and Value

- `int HasValue()` const
Return True if this node has Value and its Image. If node has values, it can be set by `SetImage()` and sometimes by `SetValue()`. Otherwise return FALSE.
- `const VString &Image()` const
Return Image of the Value stored in this node.
The same Value can have several different Images. Example: integer Value 1 can have Image 01 or H"0001". Call `Image()` only for nodes for which `HasValue()` returns True.

We return a constant reference to the image string. You cannot change the image at this node by assigning to the returned reference. Use `SetImage()` function for this purpose.

Check derived nodes description for meaning of Image and Value for a particular derived node.

- `VString Value() const`

Return Normalized Image of the Value stored in this node. Example: integer 1 has the only Normalized Image of the Value 1.

Do not call `Value()` for nodes that have no value. See specific derived node for further explanations.

- `VString IntValue() const`

Return integer Value stored in this node.

You can call `IntValue()` only for `SctIntNumConst` nodes.

- `SctErrCode SetImage(const VString &new_image, VString &err_msg)`

Set Image at this node. Derive value from the new image.

The new image is parsed and the new value is set according to the new image. If the new value cannot be parsed (has no meaning for this node), then error code is returned and error message is written to `err_msg`. If the new value was successfully parsed, `SCT_OK` is returned.

Do not call `SetImage()` for nodes that have no image/value. See specific derived node for further explanations.

- `void SetValue(const VString &new_value)`

Set Value at this node to integer `new_value`. Derive image from the value.

Currently `SetValue(int new_value)` can be called only for `SctCharString` and `SctStrConst` nodes.

- `void SetValue(int64 new_value)`

Set Value at this node to integer `new_value`. Derive image from the value.

Currently `SetValue(int new_value)` can be called only for `SctIntNumConst` node.

2.8.3 Operations on Arguments

- `int ArgNo() const`

Return number of arguments for this node. Returns non-zero values only for `SctExprOper` nodes.

- `int FixedArgNo() const`

Return the number of the fixed arguments for this node. Returns non-zero values only for `SctExprOper` nodes.

- `SctExpr *Args(int arg_ix) const`

Return pointer to argument (child) number `arg_ix` of this node.

First argument of the node has number 0, last argument of the node with N arguments has number N-1. If `arg_ix` is out of range, expression error function is called and NULL is returned. This function may be called only for `SctExprOper` nodes.

- **SctArgSlot *ArgSlots(int arg_ix) const**
Return pointer to slot of the argument number **arg_ix** of this node.
First slot of the node has number 0, last slot of the node with N slots/arguments has number N-1. If **arg_ix** is out of range, expression error function is called and NULL is returned. This function may be called only for **SctExprOper** nodes.
- **SctExpr *NextSib()**
Return next sibling of this node.
- **SctExpr *PrevSib()**
Return previous sibling of this node.
- **void SetArg(int arg_ix, SctExpr *e, int lex_ix=-1, int from_src=FALSE)**
Set argument (child) number **arg_ix** of this node to expression **e**. If **arg_ix** is out of range, expression error function is called. This function may be called only for **SctExprOper** nodes.
If **lex_ix** is not negative (by default it is -1), set lexical index of the expression **e** in this node to **lex_ix**.
If **from_src** is not FALSE, set **ArgSlots(arg_ix)->from_src** to **from_src**.
- **void SetArg(int arg_ix, SctArgSlot *slot)**
Set argument number **arg_ix** of this node to given argument slot **slot**. To be used by SCT implementation only.
- **void SetArgToList(int arg_ix, SctOper list_oper, SctExpr *e)**
Add expression **e** to the list of similar expressions. This list is at the argument number **arg_ix** of this node. This list has a list operation **list_oper**.
If the list node already exists, just add **e** to the list node as the last argument. If the list node does not exist, create it with operation **list_oper**.
- **void AddArg(SctExpr *e)**
Increase number of children of this node by one, add expression **e** to the children of this node as the new last child.
- **void InsArg(SctExpr *e, int arg_ix)**
Insert expression **e** before the child number **arg_ix** of this node. Shift all children starting with the child **arg_ix** to the right by 1. Increase number of children of this node by one.
- **void InsSibBefore(SctExpr *e)**
Insert the argument that points to **e** before this node. The parent node of this node must be a list node.
- **void InsSibAfter(SctExpr *e)**
Insert the argument that points to **e** after this node. The parent node of this node must be a list node.
- **SctExpr *DelArg(int arg_ix, int shift=-1, int create_nulls=TRUE)**
Detach child (argument) number **arg_ix** from this node. The detached child is returned.

If `shift` is `True` then the slot of the detached child is collapsed. If `shift` is `False`, then the slot of the detached child is filled with `ENULL`. If `shift` is `-1` (default value), then `shift` is computed based on this node type: for positional nodes `shift = False`, for list nodes `shift = True`.

If `create_nulls` is `True` and `SctExpr::null_nodes` is `True` then replace the removed node with `SctNullNode`. Otherwise replace the removed node with `NULL` pointer.

Please note that detached expression is not deleted but returned to the client. Client is responsible for deleting or reusing it.

- `SctArgSlot *DelArgSlot(int arg_ix, int shift=-1, int create_nulls=TRUE)`
Delete argument number `arg_ix` of this node and return its slot.
- `void MergeArgs(SctExpr *e)`
Detach all arguments from expression `e` and add them to the arguments of this node.
- `SctExpr *DelThis()`
Detach this node from the tree and return it. If parent is list node, then the slot of the detached node is deleted from the parent, If parent is positional node, then parent's pointer to this node is replaced with `ENULL`.
- `SctExpr *ReplThis(SctExpr *e)`
Detach this node from the tree and return it. The pointer to the detached node in the parent is replaced with `e`.
- `int FindArg(SctExpr *e) const`
Find expression `e` among arguments (children) of this node. Return index of the found argument, or `-1` if `e` is not among children of this node.
- `int FindMe(SctExpr *&p) const`
Find this node among the children of the parent node.
Address of parent is written to `p`, and the index of this node among parent's children is returned.
Return `-1` if this node is not found among the children of the parent node (means that the tree is corrupted).
- `int FindMe() const`
Same as above, but parent address is not reported.
- `SctArgSlot *FindMySlot() const`
Find this node among the children of the parent node. Address of slot of this node in its parent is returned.
If this node is not found among the children of the parent node or if no parent exists, return `NULL`.
- `SctExpr *Parent() const`
Return pointer to the parent of this node.

- `static int null_nodes`

Control flag: if `True`, do not allow `NULL` pointers to be children of any node. That is, translate `NULL` pointers into new `SctNullNode()` is `SetArg()` and check that there are no `NULL` pointers returned by `Args()`.

2.8.4 Links

- `int HasLink() const`

Return `True` if this node has a single `SctLink` that points to another node in the tree.

- `char *LinkTag() const`

Return the tag string that identifies this `Link`. Call this function only if `HasLink()` returns `True`.

- `SctLink &Link()`

Return the reference to the `SctLink` at this node. Call this function only if `HasLink()` returns `True`.

`Link()` return the following depending on this node:

- * `SctNameUse`
Symbol Table Element for the name used here.
- * `SctNamedObjDef`
Symbol Table Element for the top name of the Named Object defined here.
- * `SctNamedObjUse`
Definition of the Named Object referenced here.

- `int HasLinks() const`

Return `True` if this node has multiple `SctLinks` that point to another nodes in the tree.

- `char *LinksTag() const`

Return the tag string that identifies these `SctLinks`.

- `SctLinkList &Links()`

Return the reference to the list of `SctLinks` at this node.

the following is returned depending on this node:

- * `SctSymTabElt`
List of `SctNamedObjDef` nodes that refer to the name defined here.
- * `SctNamedObjDef`
List of all uses of the Named Object defined here.

2.8.5 SourcePrint-related and positions in source

- `int FromSrc() const`

Return `from_src` flag for this node.

- `void SetFromSrc(int i_from_src)`
Set `from_src` flag at this node to `i_from_src`.
- `SctExpr *ResetFromSrc()`
Reset `from_src` flag at this node and at the whole subtree that originates from this node. Return processed expression (this pointer).
- `void SetLexIx(int lex_ix)`
Set lexical index of the slot pointing to this node.
- `void RaiseComments()`
Raise the comments at the subtree that starts at this node. Call this routine before `PrettyPrint` to improve source-printing of mid-statement comments.
- `SctTxtPosn SrcPosnBeg() const`
Return start source position of this node. Specifically, return position of the first non-space non-comment source token on this tree.
- `SctTokenList &FirstTokens()`
Return reference to source tokens that start this node.
- `void RemoveSpaceTokens()`
Remove "well-behaved" space tokens inserted by `SetSlotPosn()`.

2.8.6 Tree Externalization

- `SctErrCode InspectTree(SctTokenList &err_msgs)`
Inspect the tree. Check that parent-child links are consistent, and that `Link()` and `Links()` point to the nodes found on this tree. If tree corruption is found, write error messages to `err_msgs` and return error code.
- `void PutTic(ostream &ost)`
Write out this tree to ostream `ost` in Transformer Intermediate Code form (TIC).
TIC is a way of externalizing (flattening) the tree into a platform-independent sequence of bytes that can be stored in a file or sent across the network.
- `virtual void PutTicOper(ostream &ost) const`
Write out this tree node to ostream `ost` in TIC. Used by `PutTic()`, do not use directly.
- `static int debug_tic`
Control flag: TIC output debugging level.
- `static SctNodeIndex node_ctr`
Node counter. After `PutTic()` finishes, it contains the number of nodes in the tree.
- `static SctErrCode GetTic(istream &ist, SctExpr *&ret, VString &err_msg)`
Get a TIC form of Program Tree from the input stream `ist`. Write to `ret` pointer to the Program Tree restored from the TIC form. If TIC form cannot be parsed, write error message to `err_msg` and return error code.

- virtual SctErrCode GetTicOper(istream &ist, VString &msg)
Read in this tree node from istream `ist` in TIC. Used by `GetTic()`, do not use directly.
- static long get_line_no
Input line number in the TIC input stream where `GetTic` stopped last time.

2.8.7 Errors, Debugging, and Monitoring

- void DebugPrint(ostream &ost) const
Write an indented printout of this tree to the stream `ost`. Use this function to view a tree.
Note: it is very helpful to manually call this function in debugger when you need to visualize a current state of tree that you process.
- static int tree_print_level
Level of detail while printing the tree in `DebugPrint`.
- void (*pPError)(int err_type, const SctTxtPosn &pos, const VString &s)
Pointer to user-defined SevereError- and TerminalError-processing function. If CobolTransformer Tree Expression code needs to issue Severe or Terminal Error message, this function is called.

Severe and Terminal Errors do not happen in the course of normal processing. All warnings and regular parsing and pretty-printing errors are reported as such, not as severe errors.

Severe and Terminal Errors usually point to some internal problem encountered by CobolTransformer. These errors must be immediately reported to Siber Systems.

The user-defined function pointed to by this pointer must inform the user about the Severe/Terminal Error. Then it can continue, if Severe Error was encountered, but it was not generally recommended to continue. If Terminal Error was encountered, the user function must terminate the whole application.

By default SCT-supplied function is called that simply writes severe message to `cout` and `exits`.
- static void (*pMonitor)(SctTreeEvent event, SctExpr *e1, SctExpr *e2, int n, const VString &s)
If this function pointer is not NULL, call function pointed to by `pMonitor` every time the `SctExpr` function that changes the tree is called. The called function receives arguments that describe the change that is about to occur.

2.8.8 Public Data Members

Here we describe public data members that can be freely accessed and modified.

The data members not described here are off limits to the SCT users.

- SctNodeIndex index
Index of this node. Used for tree externalization. Assigned by the node creation functions. Fully recomputed by `PutTic()` function.

- `SctUserNode *user_obj`

Pointer to User-Defined Object hanging off the node.

Some CobolTransformer users find it convenient to hang their own additional data nodes onto the `SctExpr` nodes. The `user_obj` pointer is official way to do it. You need to define preprocessor variable `SCT_USER_NODE` in order to use `user_obj` pointer.

We also suggest that you inherit your data object from `SctUserNode` class. Then the destructor of `SctExpr` node automatically calls destructor of the corresponding `*user_obj`.

- `SctTokenList tail`

Source Tokens that go after the last child of this node but before the end of this node.

2.9 Derived (Concrete) Node Classes

`SctExpr` is a base class for the following concrete node classes:

2.9.1 Constant Nodes

- `SctNullNode`

Empty Null Node. Operation `EX_NULL_NODE`. Has no image. Has no children.

If a positional `SctExpr` node does not have a child at some position then pointer at this position usually is a `NULL` pointer. However, this `NULL` pointer can be replaced with `SctNullNode`, if necessary.

- * `SctNullNode()`
Default constructor.
- * `SctNullNode(const SctNullNode &e)`
Copy constructor.

- `SctStrConst`

String Constant, or NonNumeric Literal. Operation `EX_STR_CONST`. Has no children. Contains both Image and Value of NonNumeric Literal.

Value of `SctStrConst` can have several images. For instance, string `ABC` may have images `'ABC'` and `X'414243'`.

- * `friend VString SctValueOfStrConst(const VString &image, VString &err_msg)`
Convert nonnumeric constant `image` into its value and return the value. If image cannot be parsed, write error message to `err_msg` and return null string.
- * `friend VString SctImageOfStrConst(const VString &value)`
Convert nonnumeric constant `value` into its normalized image. Return the image.
- * `SctStrConst()`
Default constructor. Construct non-numeric empty-string literal.
- * `SctStrConst(const VString &i_image)`
Construct non-numeric literal with image `i_image` and value derived from image.

- * `SctStrConst(const VString &i_image, const VString &i_value)`
Construct non-numeric literal with image `i_image` and value `i_value`).
- * `SctStrConst(const SctStrConst &e)`
Copy constructor.
- * `const VString &Image() const`
Return reference to the actual image of this string; delimiters are included, hexadecimal numbers that represent bytes are not interpreted. Example: 3-byte literal "ABC" has 5-byte image, 2-byte literal X"AABB" has 7-byte image.
- * `VString Value() const`
Return normalized image of this string. Example: normalized image for X"404142" is "@AB".
- * `SctErrCode SetImage(const VString &new_image, VString &err_msg)`
Set string image at this node to `new_image`. Write error message and return error code if new image cannot be parsed.
- * `void SetValue(const VString &new_value)`
Set Value at this node to integer `new_value`. Derive image from the value.
- * `VString image`
Data member: String Image (quotes present).
- * `VString value`
Data member: String Value (quotes stripped).

- **SctCharString**

Character String. Operation `CHAR.STRING`. Has no children. Contains both Image and Value of Character-String used in Picture clause, or any other Character-String that is not the name of the object but looks like a name. Image and Value are always the same.

- * `SctCharString()`
Default constructor. Construct empty character-string.
- * `SctCharString(const VString &i_value)`
Construct character-string with value `i_value`.
- * `SctCharString(const SctCharString &e)`
Copy constructor.
- * `const VString &Image() const`
Return reference to the actual image of this character-string.
- * `VString Value() const`
Return normalized image of this character-string. Same as actual image.
- * `SctErrCode SetImage(const VString &new_image, VString &err_msg)`
Set image and value of this character-string.
- * `void SetValue(const VString &new_value)`
Set Value at this node to integer `new_value`. Derive image from the value.
- * `VString value`
Data member: This character-string image and value.

- **SctIntNumConst**

Integer Signed Numeric constant. Operation `INT_NUM_CONST`. Has no children. Contains Value and Image of Integer Numeric Literal.

One Value may have several different Images. `SctIntNumConst` Value 10 may have the following images: 0010, +10, H'0A'.

- * `friend SctErrCode SctIntImageToValue(const VString &image, int64 &value, VString &err_msg)`
Parse decimal/hexadecimal integer `image` and place the parsed value into `value`. If parsing error occurs, write error message to `err_msg` and return error code.
- * `SctIntNumConst()`
Default constructor. Construct integer constant 0.
- * `SctIntNumConst(const VString &i_image, const int64 &i_value)`
Construct integer constant with image `i_image` and value `i_value`.
- * `SctIntNumConst(const int64 &i_value)`
Construct integer constant with value `i_value`. Use normalized image of the given value.
- * `SctIntNumConst(const SctIntNumConst &e)`
Copy constructor.
- * `const VString &Image() const`
Return actual image of this integer constant. Examples: 10, +10, or H"0A".
- * `VString Value() const`
Return normalized image of this integer constant. Examples: 10, 0, -10.
- * `VString IntValue() const`
Return integer Value stored in this node.
- * `SctErrCode SetImage(const VString &new_image, VString &err_msg)`
Set image of this integer constant. The value is parsed from the image.
- * `void SetValue(int new_value)`
Set Value at this node to integer `new_value`. Derive image from the value.
- * `VString image`
Data member: image of this integer constant.
- * `int64 value`
Data member: value of this integer constant.

- **SctRealNumConst**

Real Numeric constant. Operation `REAL_NUM_CONST`. Has no children. Contains image of Fixed or Floating Numeric Literal.

Currently only the image is stored. Value is not stored because computations on `SctRealNumConst` would be platform-specific and therefore should not be done in source-to-source tools.

- * `SctRealNumConst()`
Default constructor, Construct real numeric constant 0.0.
- * `SctRealNumConst(const VString &i_image)`
Construct real/fixed numeric constant from image `i_image`.
- * `SctRealNumConst(const SctRealNumConst &e)`

Copy constructor.

- * `const VString &Image() const`
Return actual image of this numeric constant.
- * `VString Value() const`
Return normalized image of this numeric constant. Same as actual image.
- * `SctErrCode SetImage(const VString &new_image, VString &err_msg)`
Set the image and value of this numeric constant to `new_image`.
- * `VString image`
Data member: image of this numeric constant.

2.9.2 Operation Nodes

- **SctExprOper**

Expression Operation. Has zero, one, two, or many arguments (children). All nodes other than `SctExprOper` and `SctNamedObjUse` are leaf nodes and therefore they have no children.

Every `SctExpr` node has an operation code `op` stored in it. In `SctExprOper` nodes operation code specifies the Cobol operation represented by this node. In non-operation nodes operation code is not really required, but it is present for uniformity.

Expression Operation nodes have no `Value` and no `Image`. Expression operation has no public data members.

The detailed description of all possible `SctExprOper` nodes is given in Chapter~10 that is wholly devoted to the structure of the Program Tree.

- * `SctExprOper(SctOper i_op)`
Construct operation `i_op` with default number (`SctOperArgNo(i_op)`) of `ENULL` arguments.
- * `SctExprOper(SctOper i_op, SctExpr *t1)`
Construct operation `i_op` with 1 argument `t1`.
- * `SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2)`
Construct operation `i_op` with 2 arguments `t1` and `t2`.
- * `SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3)`
Construct operation `i_op` with 3 arguments `t1`, `t2`, and `t3`.
- * `SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4)`
Construct operation `i_op` with 4 arguments.
- * `SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4, SctExpr *t5)` Construct operation `i_op` with 5 arguments.
- * `SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4, SctExpr *t5, SctExpr *t6)` Construct operation `i_op` with 6 arguments.
- * `SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4,`

- SctExpr *t5, SctExpr *t6, SctExpr *t7) Construct operation i_op with 7 arguments.
- * SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4, SctExpr *t5, SctExpr *t6, SctExpr *t7, SctExpr *t8) Construct operation i_op with 8 arguments.
- * SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4, SctExpr *t5, SctExpr *t6, SctExpr *t7, SctExpr *t8, SctExpr *t9) Construct operation i_op with 9 arguments.
- * SctExprOper(SctOper i_op, SctExpr *t1, SctExpr *t2, SctExpr *t3, SctExpr *t4, SctExpr *t5, SctExpr *t6, SctExpr *t7, SctExpr *t8, SctExpr *t9, SctExpr *t10) Construct operation i_op with 10 arguments.
- * SctExprOper(SctOper i_op, SctExpr **i_args, const int i_narg) Construct operation i_op with i_narg arguments stored in array i_args. Expressions in array i_args are consumed.
- * SctExprOper(const SctExprOper &e) Copy constructor.
- * ~SctExprOper() Destructor. Should be virtual because SctNamedObjUse inherits from it.

2.9.3 Named Object Use and Definition Nodes

- SctSymTabElt

Symbol Table Element. Operation SYMBOL_TABLE_ELT. Has no children. Contains the name stored in this Symbol Table Element. Every name used in Program Tree is represented by Symbol Table Element. SctSymTabElt's Value and Image is the Name of this Symbol Table Element.

These nodes are arranged into list with operation SYMBOL_TABLE_ELT_LIST and this list, in turn, is attached to the main program node COBOL_PROGRAM or COBOL_FILE.

- * SctSymTabElt() Default constructor.
- * SctSymTabElt(const VString &i_name, SctSymTabCat i_cat, const SctLinkList &i_refs) Create Symbol Table Element with name i_name, category i_cat, and reference list i_refs.
- * SctSymTabElt(const SctSymTabElt &e) Copy constructor. All copying is expected to happen within a tree, therefore symbol table element should not be copied at all. Moreover, since every named object definition can point to only one symbol table element, copying SctSymTabElt produces incorrect program and therefore should not be used.
- * ~SctSymTabElt() Destructor. Invalidate all SctLinks that point to this symbol table element.

- * `const VString &Image() const`
Return reference to the name in this node.
- * `VString Value() const`
Return the name stored in this node.
- * `SctErrCode SetImage(const VString &new_image, VString &)`
Set the name in this node.
- * `VString name`
Data item: user-defined word or name.
- * `SctSymTabCat cat`
Data item: Broad Category of this name: `UDC_PROC_NAME` if this name is used in Procedure Division, `UDC_DATA_NAME` if this name is used in Data Division.
Data Division and Procedure Division names should not intersect. This broad category indicates in which division the name is defined. Exact category of the Named Object appears at its definition.
- * `SctLinkList refs`
List of `SctLinks` to all references to this symbol table element in the program. `SctName-
dObjDef` and `SctNameUse` may reference this node.
- * `int complained`
Data item: True if parser already complained about this name being undefined.
- * `static int debug_symbol`
Control Flag: Symbol Table debug level

- **SctNameUse**

Single Name Use (as opposed to Qualified Name Use). Operation `NAME_USE`.

This node represents a reference to a Single Name that may be a member of Qualified Name. The Single Name is represented by link to the name's symbol table entry.

Has no children. This node does not have image/value that can be changed directly.

- * `SctNameUse()`
Default constructor.
- * `SctNameUse(const SctLink &i_ste)`
Construct `NameUse` taht point to symbol table element `i_ste`. Add a link from the symbol table element `i_ste` to the created `NameUse` automatically.
- * `SctNameUse(const SctNameUse &e)`
Copy constructor. We expect copies to be made within a tree. Therefore the copied `SctNameUse` points to the same symbol table element as the original `SctNameUse`. So we add the created `SctNameUse` to the list of references at the symbol table element.
- * `~SctNameUse()`
Destructor.
- * `const VString &Image() const`
Return reference to the name stored at the symbol table element.
- * `VString Value() const`
Return the name stored at the symbol table element.
- * `SctErrCode SetImage(const VString &new_image, VString &err_msg)`

Set the name of the object used in this node. Create Symbol Table element if new name does not exist yet.

* **SctLink ste**

Data member: SctLink to Symbol Table element for the name referenced at this node.

- **SctNamedObjDef**

Definition of the Named Object. Operation **NAMED_OBJ_DEF**.

This node serves as a point of Definition for the Named Object. All Uses of Named Object refer to the Definition Node for the object.

Definition has no name Image that can be assigned to. Instead this node refers to Symbol Table Entry that contains the name of this Named Object.

* **SctNamedObjDef()**

Default constructor.

* **SctNamedObjDef(SctSymTabCat i_cat, const SctLink &i_ste, const SctLinkList &i_uses)**

Construct NamedObjDef with category **i_cat**, name **i_ste**, and list of uses **i_uses**. Automatically add this node to the list of references at the name's symbol table element.

* **SctNamedObjDef(const SctNamedObjDef &e)**

Copy constructor. The copying is expected to happen within a single tree. The uses of the old definition are not copied to the new definition, but the new node is added to the reference list at the name's symbol table element.

* **~SctNamedObjDef()**

Destructor.

* **const VString &Image() const**

Return reference to the first name of the named object defined at this node.

* **VString Value() const**

Return the first name of the named object defined at this node.

* **SctErrCode SetImage(const VString &new_image, VString &err_msg)**

Set the name of the object defined in this node. Create Symbol Table element for the new name if necessary.

* **SctSymTabCat cat**

Data member: Exact category of the Named Object defined here.

* **SctLink ste**

Data member: SctLink to Symbol Table element for the first name of the named object defined here.

* **SctLinkList uses**

Data member: List of SctLinks to NamedObjUses pointing to this NamedObjDef.

- **SctNamedObjUse**

Named Object Use. Operation **NAMED_OBJ_USE**. Represents Qualified Name which points to the Named Object Definition.

Has no Value nor Image, but it has the only child that is **SctExprOper(OF_IN_LIST)** node to which members of the Qualified Name are attached. Each member of the Qualified Name is represented by **SctNameUse** node.

- * **SctNamedObjUse()**
Default constructor.
- * **SctNamedObjUse(SctSymTabCat i_cat, const SctLink &i_def, SctExpr *i_qn, int i_latest = FALSE)**
Construct NamedObjUse that points to the NamedObjDef node **i_def**, consists of member names **i_qn**, and has the expected broad category **i_cat**. Flag **i_latest** is used by name resolver. Add the created NamedObjUse to the list of uses at the definition **i_def**.
- * **SctNamedObjUse(const SctNamedObjUse &e)**
Copy constructor. Copying is expected to happen withing a single tree. Therefore the new NamedObjUse is added to the list of uses at the definition of the original use, if such exists.
- * **~SctNamedObjUse()**
Destructor.
- * **SctSymTabCat cat**
Data member: Expected broad Category of this named object. UDC.PROC.NAME for para-name and section-name and UDC.DATA.NAME for all other named objects.
- * **SctLink def**
Data member: SctLink to the NamedObjDef node that defines the named object reference here.
- * **int latest**
Data member: flag: if multiple definitions, use the latest definition.

2.10 Utility Functions

2.10.1 Null Nodes and Null Pointes

These are classes and functions that are closely related to **SctExpr** but they happen to be not defined inside the **SctExpr** class definition.

- **int SctIsNull(SctExpr *e)**
Return True if pointer **e** is NULL or it points to **SctNullNode**. Implemented as a C++ macro.
- **int SctIsNullList(SctExpr *e)**
Return True if pointer **e** is NULL or it points to **SctNullNode**, or it points to list node with zero children.
- **void SctNullPtrsToNullNodes(SctExpr *&e)**
In the tree **e** replace all NULL pointers to children with **SctNullNodes**. If **e** is NULL, it is also replaced.
- **void SctNullNodesToNullPtrs(SctExpr *e)**
Replace all **SctNullNodes** in the tree with NULL pointers.
- **void SctCleanOutNullsInLists(SctExpr *e)**
Clean out NULL arguments (children) of all list nodes on the tree **e**. Normally these NULL arguments should not appear at all, but they may pop up as a result of transformations.

2.10.2 List Nodes

- `SctExpr *SctListExpr(SctOper op, SctExpr *e1, SctExpr *e2)`

Construct a list node with operation `op` and arguments `e1` and `e2`. List operation `op` swallows the list operations of the same kind appearing at the arguments `e1` and `e2`.

The formal construction rules are:

$$\begin{aligned} op(op(e_1, e_2), e_3) &\Rightarrow op(e_1, e_2, e_3) \\ op(e_1, op(e_2, e_3)) &\Rightarrow op(e_1, e_2, e_3) \end{aligned}$$

- `SctExpr *SctListExpr(SctOper op, SctExpr *e)`

Same as above, but there is only one argument. The rules are exactly as above with `e2 = NULL`.

- `void SctFlattenList(SctExpr *e)`

Flatten two-level list, second level of which starts at node `e`.

2.10.3 Object Properties

- `int SctIsIdentifier(SctExpr *e)`

Return True if supplied node represents identifier. Otherwise return False.

- `int SctIsSpecialRegister(SctExpr *e)`

Return True if data item represented by `e` is a Special Register. `e` can be `SctNamedObjUse` or `SctNamedObjDef` node representing data item. Otherwise return False.

2.10.4 Picture And Usage Helpers

- `VString SctDataCatStr(SctOper cat)`

Return a 2-character string that describes data item category `cat`. One of the following is returned: "IN", "A", "N", "AN", "AE", "NE", etc.

- `SctOper SctPicStrCat(const VString &pic_str, int &length, SctTokenList &msgs)`

Return category and of the PIC string `pic_str`. Write unrolled PIC string length to `length`. Add error messages to `msgs`.

- `SctErrCode SctPicStrCategory(const VString &pic_str, VString &unrolled, SctOper &category, int &length, int &no_9s, SctExpr *context, VString &err_msg)`

Parse given picture string `pic_str`. Write the unrolled picture string to `unrolled`. Write category of the data item described by this picture string to `category`. Write the unrolled size (length) of the data item described by this picture string to `length`. Picture character `S` is not counted towards the size. Write number of 9s in this picture string to `no_9s`.

If picture string repeater contains reference to 78-level symbolic constant, use Program Tree context `context` to find the value of this symbolic constant. If `pic_str` cannot be parsed, return error code and write message to `err_msg`.

- `SctErrCode SctUnrollPicStr(const VString &pic_str, VString &out_str, SctExpr *context, VString &err_msg)`

Unroll picture string `pic_str`: substitute `X(n)` by `XXX...XX` (n times). Write the unrolled string to `out_str`.

If picture string repeater contains reference to 78-level symbolic constant, use Program Tree context `context` to find the value of this symbolic constant. If `pic_str` cannot be parsed, return error code and write message to `err_msg`.

- `void SctEffectiveUsageSyncSign(SctExpr *e, SctOper &usage, SctOper &sync, SctOper &sign, int &sign_sep_char)`

Write the effective USAGE of the DECL_DD node `e` to `usage`. If no usage is specified at this node, walk up the tree and check USAGE clauses of parent DECL_DDs. If no USAGE is found, set the effective usage to `DU_DISPLAY`.

Write the effective SYNC clause of the DECL_DD node `e` to `sync`: one of `DX_SYNC_LEFT`, `DX_SYNC_RIGHT`, or `NULL_NODE` for un-synced data item. If no SYNC clause is specified at this DECL_DD, walk up the tree and check SYNC clauses of parent DECL_DDs. If no SYNC is found, set the effective SYNC to `NULL_NODE`.

Write the effective SIGN clause of the given DECL_DD node to `sign`: one of `DX_SIGN_LEADING`, `DX_SIGN_TRAILING`, or `NULL_NODE` for unsigned data item. If no SIGN clause is specified at this DECL_DD, walk up the tree and check SIGN clauses of parent DECL_DDs. If no SIGN is found, set the effective SIGN to `NULL_NODE`. Also write TRUE to `sign_sep_char`, if SEPARATE CHARACTER is specified by the effective SIGN clause.

- `SctOper SctEffectiveUsage(SctExpr *e)`

Return effective usage of the data item `e`. If no usage is specified at this node, walk up the tree and check USAGE clauses of parent DECL_DDs. If no USAGE is found, set the effective usage to `DU_DISPLAY`.

2.10.5 Comments Manipulation

- `void SctGatherTokens(SctExpr *e, SctTokenList &tl)`

Gather all tokens from all the nodes in the tree `e` and them to token list `tl`.

- `void SctGatherComments(SctExpr *e, SctTokenList &tl)`

Gather all comment tokens from all the nodes in the tree `e` and them to token list `tl`. Usually this is done before deleting the tree at `e`, so that comments of `e` may be moved somewhere else and not just vanish.

- `void SctAddCommentBefore(SctExpr *e, const VString &comment)`

Add comment `comment` to the front of expression node `e`.

2.10.6 Descendants and Ancestors

- `int SctIsDescendant(SctExpr *e, SctOper op, SctExpr **p0 = NULL)`

Return True if there exists node `p` such that `p` has operation `op` and node `e` is a descendant of node `p`. Write address of node `p` to `*pp` if `pp` is not null. Otherwise return False.

- `SctExpr *SctFindDescendant(SctExpr *e, SctOper op)`
Find node with operation `op` among descendants of node `e`. Return `ENULL` if no such node is found.
- `SctExpr *SctFindAncestor(SctExpr *e, SctOper op)`
Find ancestor of node `e` that contains operation `op` and return address of such ancestor node. If no such ancestor exists, return `NULL`.
- `SctExpr *SctFindRoot(SctExpr *e)`
Find root of the tree to which node `e` belongs.
- `SctExpr *SctFindListAncestor(SctExpr *e)`
Find ancestor of node `e` that is a list operation. If no such ancestor exists, return `NULL`.
- `SctExpr *SctEnclosingStatement(SctExpr *e)`
Return a statement that encloses a given node `e`. By "statement" we understand expression operation that is printed on a new line.
We find statement by walking up the tree and looking for a statement node. If no statement is found, `ENULL` pointer is returned. If `e` itself is a statement, return it.
- `SctExpr *SctEnclosingSection(SctExpr *e)`
Return a section that encloses a given node. If no enclosing section is found, `ENULL` pointer is returned.
- `int SctIsLastStmtOfSentence(SctExpr *e)`
Return `TRUE` if a given statement `e` is the last statement of a sentence and all statements on the way up from the `e` to the sentence are `STMT_IF` or `EX_ELSE`.

2.10.7 Functions to Declare and Move Data Items

- `SctExpr *SctDeclDataItem(SctExpr *&def, int level, const VString &name, int unique, const VString &pic, SctExpr *value, SctExpr *prog)`
Create and return `DECL_DD` node that declares Cobol data item.
The name of the new Named Object is `name` and system ensures that it is unique if `unique` is `TRUE`. The pointer to the new Named Object definition is written to `def`.
`level` is the numeric value of the new data item. `pic` is the picture string for the new data item. `value`, if not `NULL`, is the expression for a single value in `VALUE` clause.
- `SctExpr *SctDeclConstant78(SctExpr *&def, const VString &name, SctExpr *value, SctExpr *prog)` Create `DECL_DD` for 78-level constant.
- `void SctAddDataItemToDataDiv(SctExpr *dd, int section_ix, SctOper section_oper, SctExpr *prog)`
Add Data Item Declaration `dd` which points to `DECL_DD` node to the end of the specified section of Data Division of Program `prog`. Section is specified by its index `section_ix`.

- `SctErrCode SctDeclDataItemHierarchy(SctExpr *dd_list, SctDataItemDescr *hier, int hidden, SctExpr *prog, VString &err_msg)`

Declare Record Description which is a hierarchy of Data Item Declarations. Each declaration is described by `SctDataItemDescr` taken sequentially from `hier`. Add the list of newly created data items to the `DECL_DD_LIST` node `dd_list`. If `hidden` is `True`, then hide the added declarations behind `DO_NOT_PRINT` operation. If submitted hierarchy is not correct, return error code and error message.

- `SctExpr *SctMoveStmtSingle(SctExpr *from, SctExpr *to)`

Create and return `MOVE` statement with single sending data item. This statement has `from` as a sending data item and `to` as a receiving data item.

- `SctExpr *SctShortestUniqueNamedObjUse(SctExpr *def)`

Create and return `SctNamedObjUse` node that refers to definition node `def`. The `OF_IN_LIST` list of names attached to the created `SctNamedObjUse` node is shortest possible. That is, we create a shortest name sequence that uniquely refers to a given named object.

- `SctExpr *SctShortestUniqueQualNameExpr(SctExpr *def, int ¬_unique)`

Create and return a `OF_IN_LIST` tree that represents shortest sequence of `SctNameUses` that uniquely identify Named Object with definition `def`. If the computed qualified name cannot be made unique, set `not_unique` to `TRUE`.

- `SctExpr *SctNewNamedObjDef(const VString &name, SctSymTabCat cat, int unique, SctExpr *prog)`

Create and return `SctNamedObjDef` node that defines a new Named Object.

If `unique` is `TRUE`, the new Named Object will have unique name and this unique name is based on string `name`. If `name` is already unique, just use it for the new object, otherwise add numeric suffix to it to make it unique. Symbol (Name) Table is taken from the Cobol program `prog`.

After renaming the definition, compute the Shortest Qualified Name for all the Named Objects that belong to the record declaration hierarchy that starts at the declaration for this definition. We need this because qualification of the Named Objects that are not renamed here, but that have the same First Name as `new_name` also may change due to renaming.

Also recompute qualification for all other definitions that employ the `new_name`. We need this to avoid a situation where a new name collides with existing name and the uses need have more qualification to be unique.

- `SctNamedObjDef *SctDefForNamedObjUse(SctExpr *e)`

Return `NAMED_OBJ_DEF` node that is a definition of `NAMED_OBJ_USE` node referred in `e`. Check that definition node is indeed `NAMED_OBJ_DEF`.

2.10.8 Renaming Objects

- `int SctRenameDefAndAllUses(SctExpr *def, const VString &new_name, int rollback_if_dup)`

Rename the Named Object Definition `def` and all its Uses. Return: FALSE – the new name when qualified is unique, normal completion, TRUE – the suggested `new_name` is not unique, even when fully qualified.

If the suggested `new_name` makes definition non-unique, and `rollback_if_dup` is TRUE, then definition is renamed back to the original name.

If `rollback_if_dup` is FALSE, then the new name remains. CobolTransformer will be able to distinguish between the two new duplicates because originally they were not duplicates. However, if the modified Program Tree is pretty-printed and then parsed, expect "Double Definition" error message for this Definition.

If rename was successful, then recompute the Shortest Qualified Name for all the Named Objects whose First Name coincides with `new_name`. We need to do it because qualification of other Named Objects whose First Name coincides with `new_name` is likely to change due to this renaming.

- `void SctRenameDef(SctExpr *def, const VString &new_name)`

Rename the Named Object Definition `def` to the new name `new_name`, exactly as in the function above. Assume that `rollback_if_dup` is FALSE.

- `void SctRecomputeUsesForDef(SctExpr *def)`

Compute the Shortest Qualified Name for all the Named Objects that belong to the record declaration hierarchy that starts at the declaration node for `*def`.

- `void SctRenameSymTabEltAndAllRefs(SctExpr *ste, VString &new_name)`

Rename Symbol Table Element `ste` to the `new_name`. If the new name is not unique, try another name until the uniqueness is achieved. Changing the name in SYmbol Table Element leads to automatic renaming of all references to this name.

Use this function when you need to change all reference to a certain name, irrespective of whether they are used in one definition or in many definitions. If you need to rename a name in specific definition only, use function `SctRenameDefAndAllUses()` instead.

2.10.9 Use-Definition and Symbol Table Functions

- `SctSymTabElt *SctSymTabFind(SctExpr *sym_tab, const VString &name, int *pix = NULL)`

Find user-defined word `name` in symbol table `sym_tab`. Return pointer to the element that was found and write its index to `*pix`.

If nothing was found, return NULL and write to `*pix` index of the element that would immediately follow the requested name if it was inserted into the table. This index ranges from 0 to the number of names in table.

Since this function takes about half of the parsing time, we speed it up by storing names in alphabetic order and doing binary search on the ordered names.

- `void SctSymTabAdd(SctExpr *sym_tab, SctSymTabElt *ste)`

Add Symbol Table element `ste` to Symbol table `sym_tab`. If the element being added already exists in the table, complain bitterly.

- `void SctSymTabAddElt(SctExpr *sym_tab, SctSymTabElt *ste, int ix)`
Add Symbol Table element `ste` to Symbol table `sym_tab` at position `ix`. Do not check for duplicate names.
- `SctSymTabElt *SctSymTabFindOrCreate(SctExpr *sym_tab, const VString &name)`
Find user-defined word `name` in symbol table `sym_tab`. If not found, create a new element with category `UDC_UNKNOWN`.
- `SctExpr *SctSymTabCreateUnique(const VString &i_name, SctSymTabCat cat, SctExpr *sym_tab)`
Create Symbol Table Element with new unique name `name` and category `cat`. However, if the name `name` is already used by another Named Object, CobolTransformer adds a numeric suffix to the name to make it unique. Add the new Symbol Table Element to the Symbol Table `sym_tab`.
- `void SctSymTabDeleteDef(SctExpr *def)`
Delete Name pointed to by definition `def` from its Symbol Table. Update the related link lists.
- `SctExpr *SctGetSymTab(SctExpr *e)`
Find Symbol Table of a program that encloses given node. Return `ENULL` if Symbol Table cannot be found.
- `SctExpr *SctSymTabForDef(SctExpr *def)`
Return pointer to Symbol Table element for a name defined in definition `def`.
- `SctExpr *SctNameUseForDef(SctExpr *def)`
Create `SctNameUse` node that refers to a name pointed to by definition `def`.
- `SctExpr *SctNamedObjUseForName(const VString &name, SctExpr *context)`
Create `SctNamedObjUse` node that refers to an object named `name`. Context node `context` is used to find a program in which names are looked up.
- `SctExpr *SctFullQualNameExpr(SctExpr *def)`
Create and return `OF_IN_LIST` list expression that represents a Fully Qualified Name for Named Object Definition `def`.
- `VString SctFullQualNameStr(SctExpr *def)`
Return Fully Qualified Name for Named Object Definition `def` in "a OF b OF c" notation.
- `VString SctFullDottedQualNameStr(SctExpr *def, char sep='.')`
Return Fully Qualified Name for Named Object Definition `def` in "c.b.a" notation.
- `void SctRestrictDefs(SctLinkList &e1defs, const SctLinkList &e2defs)`
Restrict list of definitions in `e1defs` to the definitions that are among the definitions in `e2defs`. That is, for all `e1def` in `e1defs`: If not exists `e2def` in `e2defs` s.t. `e1def` OF `e2def` is possible (`e2def` path is subset of `e1def` path), then delete `e1def` from `e1defs`.

- `SctExpr *SctParentDeclaration(SctExpr *e)`
Return Parent Declaration for the Named Object Declaration (not Definition) `e`. Parent-child relationship for Declarations are mandated by Record Description Entry hierarchy. Return NULL if this declaration is a top level one (that is, it has no parent declaration).
- `SctExpr *SctNamedObjDeclForDef(SctExpr *e)`
Return Declaration node for the definition node `e`.
- `SctExpr *SctNamedObjDefForDecl(SctExpr *decl)`
Return pointer to the existing `SctNamedObjDef` node that defines Named Object declared in Declaration Node `decl`. Return NULL for nodes that are not Declaration nodes.
- `VString SctQualNameStr(SctExpr *qual_name)`
Return a string that represents Qualified Name stored in `OF_IN_LIST` list expression `qual_name` in "a OF b OF c" notation.
- `VString SctDottedQualNameStr(SctExpr *qual_name, char sep='.')`
Return a string that represents Qualified Name stored in `OF_IN_LIST` list expression `qual_name` in "c.b.a" notation.
- `SymTabQualUse(SctExpr *qual_name, SctSymTabCat cat, int latest)`
Create `SctNamedObjUse` node that represents an `OF_IN_LIST` Qualified Reference `qual_name` to the Named Object. The category suggested by the name use is `cat`.
- `SctExpr *SctSymTabNonQualUse(SctExpr *ste, SctSymTabCat cat, SctTokenList *ptokens = NULL)` Create `SctNamedObjUse` node that represents use of the Named Object of the category `cat` specified by non-qualified name `ste`.
- `void SctResolveNamedObjUses(SctExpr *e)`
Resolve `NAMED_OBJ_USE` nodes in the subtree `e`. That is, find definitions for this use and correspondingly write `def` link in this `NAMED_OBJ_USE` node.
Use this function only for after=parsing name resolution.

2.10.10 Miscellaneous

- `void SctDelete(SctExpr *&e)`
Delete tree at `e` and nullify pointer `e`.

2.10.11 Tree Walker Functions

- `SctWalkOrder WalkTheTree(SctExpr *const e, SctExpr *const root, SctWalkOrder (*pf)(SctExpr *const e, SctExpr *const root))`
Walk all nodes of the tree `e` and call user-defined function `*pf` at every visited node. The processing function `*pf` receives the pointer to the node being visited as the parameter `e`. It will receive another parameter `root` that is passed all the way through from the original call to `WalkTheTree`.

Function ***pf** can change members of the current node and its children, it can even replace and/or delete the node ***e** itself, but if it does it, the function ***pf** must detach node ***e** from the rest of the tree.

Function ***pf** must return the walking order code (enum `SctWalkOrder`):

- * **SCT_WALK_ARGS**
Continue walking, walking the arguments (children) and siblings of this node.
- * **SCT_WALK_SIBS**
Do not walk the children of this node, but walk the siblings of this node.
- * **SCT_WALK_STOP**
Stop walking completely, escape all the way up to the root. After returning this code function ***pf** will not be called again.

The function `WalkTheTree` returns walking order code passed to it by the (***pf**).

- `SctWalkOrder WalkTheTreeAdvStart(SctExpr *const e, void *arg_block, char *tag, SctWalkOrder (*p_pre)(SctExpr *const e, SctExpr *const root, void *arg_block), SctWalkOrder (*p_post)(SctExpr *const e, SctExpr *const root, void *arg_block), SctWalkOrder (*p_tokens)(SctTokenList &tl, SctExpr *const e, SctExpr *const root, void *arg_block))`

More sophisticated tree walker.

Walk the tree **e** and call the following functions at every visited node:

Function ***p_pre** is called before walking the arguments of the node,
function ***p_post** is called after walking the arguments of the node,
function ***p_tokens** is called for every token list attached to every tree node on the walked tree. Argument block ***arg_block** is used to transfer any arguments from root caller to the processing function.

Functions ***p_pre** and ***p_post** can change the current node, they can even replace and/or delete the node ***e** itself, but if they do it, they must disconnect ***e** from the tree. Function ***p_tokens** receives token list **tl** as its first argument.

In addition to **e** and **root** arguments the processing function ***pf** receives argument **s** that point to the slot of the node **e**.

Functions ***p_pre** and ***p_post** return walking order code.

2.11 Length and Offset Computation

2.11.1 SctAlignInfo: Alignment information

CobolTransformer can compute data item lengths and offsets for various operating systems and dialects which use different memory alignment principles and have different byte sizes for basic Cobol types.

`SctAlignInfo` structure encodes this system-dependent information. The following public members are available:

- `SctAlignInfo()`
Default constructor.

- **int alignment**

Type of alignment (storage mode) used in the given system. Can be `SCT_ALIGNMENT_BYTE` or `SCT_ALIGNMENT_WORD`.

In byte storage mode size of binary items depends on the item `PICTURE` and it can be any number of bytes from 1 to 8. In word storage mode size of binary item can be 2, 4, or 8 bytes.

Default is byte alignment (byte storage mode).

- **int sign_is_trailing_separate**

Flag: Is `SIGN` for numeric items a trailing separate character.

Default: False.

- **int pointer_size**

Size of the `POINTER` data item. Depends on particular cobol system/CPU architecture. Default is 4.

- **int proc_pointer_size**

Size of the `PROCEDURE-POINTER` data item. Depends on particular cobol system/CPU architecture. Default is 4.

- **int index_size**

Size of the `INDEX` data item. Usually it is equal to `BINARY` or `POINTER` size. Default is 4.

- **int unfold_flex_arrays**

How to treat `DEPENDING ON` clauses. If `unfold_flex_arrays` is On, then size of table with `DEPENDING ON` is computed as difference between upper and lower bound on indices. If this is Off, then size table with `DEPENDING ON` is undefined.

2.11.2 Functions

The following functions are available:

- `SctErrCode SctCalcLengthsOffsetsValues(SctExpr *e, SctTokenList &err_msgs, int lengths_loc_offsets = TRUE, int global_offsets = TRUE, int constant_values = TRUE, const SctAlignInfo &placing = SctAlignInfo())`

Compute the following info: (1) for every data item compute length and local offsets, if `lengths_loc_offsets` is True, (2) for every data item compute global offsets, if `global_offsets` is True, (3) for every 78-level constant entry compute its value, if `constant_values` is True.

Platform-specific alignment information is given in the parameter `placing`.

Note: if a certain computation is requested, it will be performed by the system. However, if a certain other computation is not requested, it still may have to be done to enable another computation that was explicitly requested.

Example: only `constant_value` is requested, others are not. Program has `VALUE START` and `VALUE LENGTH` constructs, presence of these constructs automatically invokes computation of lengths and offsets, even though lengths and offsets computation was not requested by the user.

- `void SctPrintLengthsOffsets(ostream &ost, SctExpr *e)`
Print length and offset report for all data items declared in tree `e`. Report is printed to ostream `ost`.
- `SctExpr *SctCalcValueExpr(SctExpr *e)`
Calculate value represented by constant expression `*e`. Returns expression node that represents calculated entry. Destroy the returned expression if you do not use it. `ENULL` is returned when this expression refers to undefined constants (as in `START OF/LENGTH/NEXT` of yet unprocessed data).

2.12 SctLink: Externalizable Pointer

2.12.1 SctLink class

Class `SctLink` points to a node in the tree, which is not a child of this node. `SctLink` contains integer index of the pointed to node, so that externalization of such links can be performed.

The following public members are available:

- `SctLink()`
Default constructor.
- `SctLink(SctExpr *i_expr)`
Constructor: specify pointed-to node.
- `SctLink(SctExpr *i_expr, SctNodeIndex i_index)`
Constructor: specify pointed-to node and its index.
- `SctLink(const SctLink &e)`
Copy Constructor.
- `~SctLink()`
Destructor.
- `friend ostream &operator<<(ostream &ost, const SctLink &l)`
Print out link `l` in human-readable form to ostream `ost`.
- `void PutTic(ostream &ost) const`
Print this link in TIC form to the ostream `ost`.
- `SctErrCode GetTic(istream &ist, VString &err_msg)`
Get this link from TIC stream `ist`.
- `static SctLink null`
Static Null link constant.
- `SctExpr *expr`
Expression node that this link points to.
- `SctNodeIndex index`
Index of the expression node that this link points to.

2.12.2 SctLinkedList and SctLinkedListElt classes

Class `SctLinkedListElt` represents a single element of this list. It contains the link itself and pointer to the next list element. Class `SctLinkedList` represents the whole list. Most operations are performed on the whole list, not on list element.

Class `SctLinkedListElt` has the following members:

- `SctLinkedListElt(const SctLink &l, SctLinkedListElt *i_next)`
Construct link list element with link `l` and next list element `i_next`.
- `~SctLinkedListElt()`
Destruct only this link list element.
- `SctLink link`
Link in this list element (payload).
- `SctLinkedListElt *next`
Pointer to the next link list element.

Classes `SctLinkedListElt` and `SctLinkedList` are used to represent a list of `SctLink` elements. Class `SctLinkedList` is the list itself. It has the following members:

- `SctLinkedList()`
Construct an empty list.
- `SctLinkedList(const SctLinkedList &l)`
Copy constructor.
- `SctLinkedList &operator = (const SctLinkedList &l)`
Assignment operator.
- `SctLinkedList Clear()`
Nullify this list.
- `~SctLinkedList()`
Fully destroy this list and all its elements.
- `int IsNull() const`
Return `True` if this is empty list. Otherwise return `False`.
- `int Length() const`
Return number of elements in this list.
- `void AddElt(const SctLink &l)`
Add element containing link `l` to this list as the last element.
- `void DelElt(SctExpr *e)`
Delete element that has link with pointer to expr `e` from this link list.
- `SctLink &Elts(int i)`
Return reference to `SctLink` in element number `i` of this link list. If there is no element with such number, return reference to null link.

- `friend ostream &operator <<(ostream &ost, const SctLinkList &l)`
Print link list to ostream `ost` in human-readable form.
- `void PutTic(ostream &ost) const`
Write this link list to ostream `ost` in TIC form.
- `SctErrCode GetTic(istream &ist, VString &err_msg)`
Get a SctLinkList from istream `ist` in TIC form. Return error code and write error message to `err_msg` if TIC cannot be parsed.
- `static SctLinkList null`
Empty link list constant.
- `SctLinkListElt *head`
Pointer to the first element of this link list.

2.13 Miscellaneous Conversion Functions

- `SctErrCode SctTurnExprIntoComments(SctOper new_op, SctExpr *e, SctPrettyPrinter &pprinter, SctTokenList &msgs)`
Turn expression `e` into a sequence of comments using pretty-print `pprinter`. These comments are attached to the new node with operation `new_op` that replaces the node `e`. If error occurs, add error messages to `msgs` and return error code.
- `SctExpr *SctFindNodeByPositionInSource(const SctTxtPosn &posn, SctExpr *e0, int exact, SctExpr *&found_e, SctTxtPosn &found_posn)` In expression tree `e0` find a node that has text position `posn`. If `exact` is False, return node that has position closest to `posn`. If `exact` is True, return ENULL if node with such position is not found. If column numbers of two positions are different but the rest is the same, this is an exact match for purposes of this function. Both for exact and inexact search we write to `found_e` address of the node that got closest to `posn` and to `found_posn` this closest position.

2.14 OPTIONAL CLASSES START HERE

All classes and functions from here to the end of this Chapter are experimental and optional, and as such, are not officially supported.

The optional classes and function provide a number of conveniences that simplify writing converters using CobolTransformer, but add-ons are not required for CobolTransformer to operate properly.

2.15 Expression Debug Facilities

Functions to print diagnostic messages and make assertions.

- `const char *GetExpressionName (const SctExpr *e)`
Return operation name of `e` or "NULL" if `e` is NULL.

- `void Assert (int True, const char *msg = "Assertion failed")`
Assert condition `True` is satisfied. Run `SctTransform::pError()` otherwise.

2.16 Bag And Map Container Classes

2.16.1 `template <class T> class Bag`

Bag is an unordered collection of items of type `T`.

- `Bag()`
Constructor. Takes no parameters.
- `Register (T a)`
Add item `a` to bag.
- `Unregister (T a)`
Remove item `a` from bag.
- `FindItem (T a)`
Return internal index of item `a` or -1 if `a` is not in bag.
- `IsRegistered (T a)`
Check if `a` is in bag.
- `Reset()`
Remove all items from bag.
- `operator[] (int i)`
Return bag item with internal index `i`.
- `Count()`
Return number of items in bag.

2.16.2 `template<class T, class R> class Map`

Map is a mapper of items of type `T` to items of type `R`.

- `Put (T index, R value)`
Specify that item `index` should be mapped to `value`.
- `Get (T index)`
Return the item `index` is mapped to.
- `IsMappable (T index)`
Check if item `index` can be mapped.
- `Reset()`
Clear mapping table.

- **Count()**
Return number of items in table.
- **T GetKey (int n)**
Return a table key for table row **n**.
- **R GetValue (int n)**
Return value stored in a table row number **n**.

2.17 Iterators

- **ForAllListItems(item, list) / ForAllListItemsEnd**
These macros facilitate enumeration of list items.
- **ForAllUses(use, def) / ForAllUsesEnd**
These macros facilitate enumeration of uses of a **SctNamedObjDef**.

2.18 Expression Tree Manipulation Functions

Low level functions that facilitate working with expression trees.

- **SctExpr *GetArgOrCreatelt (SctExpr *e, int idx, SctOper arg_oper)**
Return argument number **idx** of expression **e**. If argument **idx** is **NULL**, set it to the new expression with operation **arg_oper**.
- **SctExpr *GetDefForUse (const SctExpr *e)**
Check if **e** is a **SctNamedObjUse** and return its definition.
- **int IsInSlot (SctExpr *e, SctOper op, int idx)**
Check if **e** occupies an argument slot number **idx** of an expression with operation **op**.
- **int ListIsEmpty (SctExpr *e)**
Return **TRUE** if list is empty, **FALSE** otherwise.
- **int IsLastChild (SctExpr *e)**
Return **TRUE** if **e** is the last child of its parent.
- **SctExpr *GetFirstArg (SctExpr *e)**
Return the first argument of expression **e** or **NULL** if **e** has no arguments.
- **SctExpr *GetLastArg (SctExpr *e)**
Return the last argument of expression **e** or **NULL** if **e** has no arguments.
- **SctExpr *GetNextInList (SctExpr *e)**
Return the expression that goes after **e** in list of arguments of parent of **e** or **NULL** if **e** is last argument.

- **SctExpr *GetPrevInList (SctExpr *e)**
Return the expression that goes before **e** in list of arguments of parent of **e** or NULL if **e** is first argument.
- **void InsertAfter (SctExpr *base, SctExpr *item)**
Insert **item** after **base** into list of arguments of parent of **base**.
- **void InsertBefore (SctExpr *base, SctExpr *item)**
Insert **item** before **base** into list of arguments of parent of **base**.
- **void InsertListBefore (SctExpr *base, SctExpr *list)**
Insert all items in **list** before **base** into list of arguments of parent of **base**.
- **void InsertListAfter (SctExpr *base, SctExpr *list)**
Insert all items in **list** after **base** into list of arguments of parent of **base**.
- **void MergeListToList (SctExpr *dst, SctExpr *src)**
Merge list **src** to list **dst**.
- **void DetachAndDestroyForever (SctExpr *e)**
Detach **e** from program tree and revoke memory occupied by **e**.
- **void ReplaceAndDestroyForever (SctExpr *e1, SctExpr *e2)**
Replace **e1** by **e2** in program tree and revoke memory occupied by **e1**.
- **void ReplaceWithListAndDestroyForever (SctExpr *e1, SctExpr *e2)**
Replace **e1** by sequence of items of list **e2** in program tree and revoke memory occupied by **e1**.
- **void ReplaceExpressionWithComments (SctExpr *e)**
Turn expression **e** into comments. Comments replace **e** in program tree.

2.19 Expression Classification Functions

Functions describing classes of particular expressions representing Cobol programs.

- **int IsIOStatement (SctExpr *e)**
Check if **e** is one of Cobol IO statements.
- **int IsExceptionClause (SctExpr *e)**
Check if **e** is an exception clause of a Cobol statement.
- **int IsOwnerOfLinkageField (SctExpr *e)**
Check if **e** has MF linkage clause. This clause is present in CALL and ENTRY statements and PROCEDURE DIVISION header.
- **int IsLiteral (SctExpr *e)**
Check if **e** is a literal.

- `int IsIdentifier (SctExpr *e)`
Check if `e` is an identifier.
- `int IsComparison (SctExpr *e)`
Check if `e` is an operation of arithmetic comparison.
- `int IsSpecialValueItem (SctExpr *e)`
Check if `e` is a special kind of VALUE specification allowed in MF: VALUE START OF, VALUE LENGTH OF, VALUE NEXT.
- `int IsFileNameUse (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying some file.
- `int IsAlnumLiteral (SctExpr *e)`
Check if `e` is an alphanumeric literal. Strings of digits are considered to be numeric, not alphanumeric literals.
- `int IsAlnumLiteralStrict (SctExpr *e)`
Check if `e` is an alphanumeric literal. Strings of digits are considered to be alphanumeric literals.
- `int IsUseOfDisplayDataItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying data item with usage DISPLAY.
- `int IsUseOfIndexDataItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying data item with usage INDEX.
- `int IsUseOfTableIndexItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying table index.
- `int IsUseOfBinaryDataItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying data item with one of binary usages: BINARY, COMP, COMP5 etc.
- `int IsUseOfAlnumDataItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying alphanumeric data item.
- `int IsUseOfNumericDataItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying numeric data item.
- `int IsUseOfNumericOrNumericEditedDataItem (SctExpr *e)`
Check if `e` is a `SctNamedObjUse` identifying numeric edited data item.
- `int IsDataEntryWithSubordinates (SctExpr *e)`
Check if `e` is a DECL_DD of a group data item.
- `int IsReportWriterFileNameDef (SctExpr *e)`
Check if `e` is a `SctNamedObjDef` identifying a file defined in REPORT WRITER section.

2.20 Cobol Program Navigation Functions

- SctExpr *GetProcessedProgram()
Return root expression of processed Cobol program.
- SctExpr *GetProcedureDivision()
Return PROCEDURE DIVISION of processed Cobol program.
- SctExpr *GetDataDivision()
Return DATA DIVISION of processed Cobol program.
- SctExpr *GetEnvironmenDivision()
Return ENVIRONMENT DIVISION of processed Cobol program.
- SctExpr *GetWSSection()
Return WORKING-STORAGE SECTION of processed Cobol program.
- SctExpr *GetFileSection()
Return FILE SECTION of processed Cobol program.
- SctExpr *GetScreenSection()
Return SCREEN SECTION of processed Cobol program.
- SctExpr *GetLinkageSection()
Return LINKAGE SECTION of processed Cobol program.
- SctExpr *GetWSSectionList()
Return list of top-level entries in WORKING-STORAGE SECTION of processed Cobol program.
- SctExpr *GetLinkageSectionList()
Return list of top-level entries in LINKAGE SECTION of processed Cobol program.
- SctExpr *GetScreenSectionList()
Return list of top-level entries in SCREEN SECTION of processed Cobol program.
- SctExpr *GetConfigurationSection()
Return CONFIGURATION SECTION of processed Cobol program.
- SctExpr *GetSpecialNamesParagraph()
Return SPECIAL-NAMES paragraph of processed Cobol program.
- SctExpr *GetSNFunctionSwitchList()
Return list of function switches in SPECIAL-NAMES paragraph of processed Cobol program.

2.21 Miscellaneous utility functions

- `int GetDataItemLength (SctExpr *e)`
Check if `e` is a data entry with correct hidden LENGTH clause and return its numeric value.
- `SctOper GetPictureCategory (SctExpr *pic)`
Return category of a picture string stored in `pic`.
- `SctExpr *SctGetFCFromFD (SctExpr *fd)`
Return File Control Entry corresponding to the given File Description Entry.
- `SctExpr *GetDataEntryForUse (SctExpr *e)`
Return Data Description Entry for which `SctNamedObjUse e` is an identifier.
- `SctExpr *GetFileNameDefFromIOStatement (SctExpr *e)`
Scrutinize Cobol IO statement `e` and return the file this statement is acting upon.
- `SctExpr *GetFileStatusDefFromFileNameDef (SctExpr *e)`
Return `SctNamedObjDef` of file status data item for file `e`.
- `SctExpr *GetRedefinedDefFromDef (SctExpr *e)`
Return `SctNamedObjDef` of data entry which is redefined by `e`.
- `SctExpr *GetParentGroupItemDefForDef (SctExpr *def)`
Return `SctNamedObjDef` of parent data entry group of `def` or NULL if `def` is a top-level data entry.
- `SctExpr *GetNthSubordinate (SctExpr *e, int n)`
Return `SctNamedObjDef` of subordinate data item with number `n`.
- `SctExpr *GetNextStatement (SctExpr *e)`
Return next statement in PROCEDURE DIVISION for statement `e`.
- `SctExpr *GetPrevStatement (SctExpr *e)`
Return previous statement in PROCEDURE DIVISION for statement `e`.
- `SctExpr *CreateNamedObjDef (const VString &name, SctSymTabCat cat)`
Return newly created `SctNamedObjDef` of given name and category.
- `SctExpr *SctCreateDataEntry (int level, const VString &name, const VString &pic = "", SctExpr *usage_type = 0)`
Return newly created Data Description Entry with given level, name, picture and usage.
- `SctExpr *SctCreateConditionNameEntry (const VString &name, SctExpr *value)`
Return newly created Condition Name Entry with given name and value.
- `SctExpr *CreateDD (int level, const VString &name, SctExpr *pic, SctExpr *usage, SctExpr *sub_list)`
Create DECL_DD expression and its Data Description Entry.

- `void ProvideEndStatementDelimiter (SctExpr *e)`
Ensure that statement `e` has END-STATEMENT delimiter.
- `void SplitStatementWithItemToListOfStatements (SctExpr *e, int arg_to_split)`
Split a statement that contains a list of similar clauses to several statements, each having only single clause. e.g. split OPEN I-O F1 INPUT F2 to OPEN I-O F1 OPEN INPUT F2.

2.22 Predicate and LightweightConversion classes

Abstract classes `Predicate` and `LightweightConversion` provide a means to create context closures for two distinct classes of functions: predicate functions that check if a tree node satisfies a condition; and conversion functions that are applied to a tree node and perform certain actions. These classes are specifically designed to be used together with tree node enumeration functions like `IterateUses/FindUse` and other similar functions declared below.

- `class Predicate`
Abstract class with the only member function `int IsSatisfied (const SctExpr *) const`. Its derivatives are supposed to define implementation of this member function together with some constant context necessary for its evaluation.
- `class LightweightConversion`
Abstract class with the only member function `void ApplyTo (SctExpr *)`. Its derivatives are supposed to define implementation of this member function together with some context necessary for its evaluation.
- `*FindAncestorThatIs (const SctExpr *e, const Predicate &p)`
Return ancestor of `e` that satisfies predicate `p`.
- `*FindDescendantThatIs (const SctExpr *e, const Predicate &p)`
Return descendant of `e` that satisfies predicate `p`.
- `*FindUseThatIs (const SctExpr *e, const Predicate &p)`
Return use of `SctNamedObjDef e` that satisfies predicate `p`.
- `IterateUsesWithUpdate (SctExpr *e, LightweightConversion &cvt)`
Enumerate uses of `SctNamedObjDef e` applying conversion `cvt` to them. Members of `cvt` are updated during conversion and can be read after the call.
- `IterateListItemsWithUpdate (SctExpr *list, LightweightConversion &cvt)`
Enumerate items of list `e` applying conversion `cvt` to them. Members of `cvt` are updated during conversion and can be read after the call.
- `IterateSubordinateDataItemsWithUpdate (SctExpr *e, LightweightConversion &cvt)`
Enumerate data items subordinate to `e` applying conversion `cvt` to them. Members of `cvt` are updated during conversion and can be read after the call.
- `IterateDescendantsWithUpdate (SctExpr *e, LightweightConversion &cvt)`
Enumerate descendants of `e` applying conversion `cvt` to them. Members of `cvt` are updated during conversion and can be read after the call.

- IterateUses (SctExpr *e, LightweightConversion &cvt)
Enumerate uses of SctNamedObjDef e applying conversion cvt to them. Members of cvt are not modified.
- IterateListItems (SctExpr *list, LightweightConversion &cvt)
Enumerate items of list e applying conversion cvt to them. Members of cvt are not modified.
- IterateSubordinateDataItems (SctExpr *e, LightweightConversion &cvt)
Enumerate data items subordinate to e applying conversion cvt to them. Members of cvt are not modified.
- IterateDescendants (SctExpr *e, LightweightConversion &cvt)
Enumerate descendants of e applying conversion cvt to them. Members of cvt are not modified.
- *GetDeclarativesErrorHandlerFor (SctExpr *e)
Return SctNamedObjDef of declarative section for the file with SctNamedObjDef e or NULL if the file does not have such section.
- int HasAncestorThatIs (const SctExpr *e, const Predicate &p)
Check if e has ancestor that satisfies predicate p.
- int HasDescendantThatIs (const SctExpr *e, const Predicate &p)
Check if e has descendant that satisfies predicate p.
- int HasUseThatIs (const SctExpr *e, const Predicate &p)
Check if SctNamedObjDef e has use that satisfies predicate p.
- FunctionInstance (PredicateFunction *f)
Predicate: f applied to expression gives TRUE.
- IsUseOf (const SctExpr *def)
Predicate: expression is a SctNamedObjUse for def.
- IsDefWithName (const VString &name)
Predicate: expression is a SctNamedObjDef with image name.
- IsExprWithID (SctOper ID)
Predicate: expression has operation number ID.
- IsTheExpr (SctExpr *e)
Predicate: expression is the expression e.
- IsSonOfExprThatIs (const Predicate &p)
Predicate: expression is a son of expression that satisfies p.
- IsDescendantOfExprThatIs (const Predicate &p)
Predicate: expression is a descendant of expression that satisfies p.

- `SctExpr *FindAncestorThatIs (SctExpr *e, PredicateFunction *f)`
Return ancestor of `e` that satisfies condition `f`.
- `SctExpr *FindAncestorWithID (SctExpr *e, SctOper ID)`
Return ancestor of `e` with operation `ID`.
- `SctExpr *FindDescendantThatIs (SctExpr *e, PredicateFunction *f)`
Return descendant of `e` that satisfies condition `f`.
- `SctExpr *FindDescendantWithID (SctExpr *e, SctOper ID)`
Return descendant of `e` with operation `ID`.
- `int IsDescendantOf (SctExpr *e, SctOper ID)`
Check if `e` is a descendant of expression with operation `ID`.
- `int IsDescendantOf (SctExpr *e, SctExpr *ancestor)`
Check if expression `e` is a descendant of expression `ancestor`.
- `int HasAncestorWithID (SctExpr *e, SctOper ID)`
Check if `e` has ancestor with operation `ID`.
- `int HasDescendantWithID (SctExpr *e, SctOper ID)`
Check if `e` has descendant with operation `ID`.
- `SctExpr *FindAncestorWithParentID (SctExpr *e, SctOper ID)`
Find ancestor of `e` whose parent has operation `ID`.
- `ReplaceWithUseOfConversion (SctExpr *def)`
LightweightConversion: Replace expression with a newly created use of `def`.
- `CommentStatementsConversion()`
LightweightConversion: Comment out expression.
- `IsUseAfterFor (const SctExpr *file_name_def)`
Predicate: expression is a `USE AFTER` statement for a file with `SctNamedObjDef file_name_def`.
- `IsDefOfDeclarativeSectionFor (const SctExpr *file_name_def)`
Predicate: expression is a declarative section for a file with `SctNamedObjDef file_name_def`.
- `IsRedefinedByOrRedefines (const SctExpr *def)`
Predicate: expression is a `SctNamedObjDef` for a data item that either redefines or is redefined by a data item with `SctNamedObjDef def`.

2.23 OPTIONAL CLASSES END HERE

This is the end of un-supported Optional Classes and Functions.

Chapter 3

Common Utility Classes

3.1 Class VString

Most of the strings used in CobolTransformer are represented using class `VString`.

3.1.1 VString class

This class has the following public members:

- * `VString()`
Default constructor constructs null string.
- * `VString(char c)`
Construct string consisting of one character `c`.
- * `VString(const char *s)`
Construct string that is a copy of null-terminated C string `s`.
- * `VString(const char *s, int i_len)`
Construct string that consists of the first `i_len` bytes of the null-terminated C string `s`.
- * `VString(ostringstream &sst)`
Construct string from the filled `ostringstream`.
Finish and freeze the stream, and reuse its `char *` array.
- * `VString(FastOstringstream &sst)`
Construct string from the filled `ostringstream`.
Finish and freeze the stream, and reuse its `char *` array.
- * `VString(const VString &vs)`
Copy constructor.
- * `VString &operator = (const VString &s)`
Assignment operator.
- * `void MoveFrom(VString &s)`
Assign this string from the string `s`, taking away the string `s` contents and nullifying the string `s`.
- * `VString &operator += (const VString &s)`
Replace this string with concatenation of this string and string `s`.

- * `~VString()`
Destructor.
- * `static VString null`
Null string constant.
- * `int IsNull(void) const`
Return True if this string is a null string.
- * `int Length(void) const`
Return length of this string.
- * `const char *Pstr() const`
Return const pointer to `char *` string sitting inside this string.
- * `char *NcPstr() const`
Return pointer to non-sconst `char *` string sitting inside this string.
- * `char &operator [] (int i) const`
Return reference to *i*-th character of this string.
- * `VString Substr(int start, int len) const`
Return substring of this string. The substring starts at the position `start` and is `len` bytes long. If position `start+len` is beyond the string length, then `len` is decreased to `Length()-start`.
Return substring of this string that goes till character `c`.
- * `int HasChar(char c) const`
Return True if this string contains character `c`.
- * `int StartsWith(const VString &starter) const`
Return True if this string starts with the string `starter`.
Return position of the first occurrence of character `c` in this string. Return -1 if there are no such occurrences.
Return number of occurrences of a given char in this string.
- * `int OnlyHasChars(const char *char_list) const`
Return True if this string contains any of the characters only from the `char_list`.
- * `int ProjectedLength(const VString &alphabet)`
ProjectedLength: compute length of this string after restricting it to `alphabet`.
- * `int.IsAnyOf(const char *str_list) const`
Return True if this string is any of the strings contained in `str_list`. Strings in `str_list` are separated from each other by binary zeros. Example of `str_list`: "MF\OIBM\OFSC".
- * `int StrIndex(const VString &ss) const`
Return position of substring `ss` in this string. If substring not found, return (-1).
- * `void Replace(const char from, const char to)`
In this string, replace all characters that have code `from` with characters that have code `to`.
- * `void Remove(const VString &rm_list)`
In this string, remove all characters that are present in the string `rm_list`.
Trim character `b` at the start of this string and character `e` at the end of this string.
- * `VString Trim() const`
Trim white space from beginning and end of the string. Returned trimmed string.
- * `friend VString operator + (const VString &s1, const VString &s2)`
Return concatenation of strings `s1` and `s2`.

- * `friend int SctVStringCmp(const VString &s1, const VString &s2)`
Return 0, if string `s1` is equal to string `s2`, -1, if `s1` is less than `s2` and +1 if `s1` is more than `s2`.
- * `friend int operator == (const VString &s1, const VString &s2)`
Return True, if string `s1` is equal to string `s2`.
- * `friend int operator == (const VString &s1, const char *s2)`
Return True, if string `s1` is equal string `s2`.
- * `friend int operator != (const VString &s1, const VString &s2)`
Return True, if string `s1` is not equal to string `s2`.
- * `friend int operator != (const VString &s1, const char *s2)`
Return True, if string `s1` is not equal to string `s2`.
- * `friend ostream& operator << (ostream &ost, const VString &s)`
Print string `s` to the ostream `ost` in human-readable form.
- * `void PutTic(ostream &ost) const`
Write this string to the ostream `ost` in TIC form with length specifier.
- * `SctErrCode GetTic(istream &ist, VString &msg)`
Read this string from the istream `ist` in TIC form with length specifier. If string cannot be read, write error message to `msg` and return error code.
- * `void PutTic(ostream &ost, char delim) const`
Write this string to the ostream `ost` in TIC form delimited by delimiter `delim`. String is written out in printable format that contains no invisible characters and can be viewed in conventional ASCII editor.
- * `SctErrCode GetTic(istream &ist, char delim, VString &msg)`
Read this string from the istream `ist` in TIC form delimited by delimiter `delim`. String representation is a printable format that contains no invisible characters and can be viewed in conventional ASCII editor. If string cannot be read, write error message to `msg` and return error code.
- * `friend VString vfmt(const char *fmt, ...)`
Sprintf the arguments to the buffer using the format string `fmt`. Return the printed to buffer as VString.

3.1.2 Case conversions

- * `VString SctToUpper(const VString &s)`
Convert string `s` to upper-case and return the result.
- * `int SctToUpper(int c)`
Convert character `c` to upper-case and return the result.
- * `VString SctToLower(const VString &s)`
Convert string `s` to lower-case and return the result.
- * `int SctToLower(int c)`
Convert character `c` to lower-case and return the result.

3.1.3 Code Page conversions

- * `void SctConvertCodePage(const VString &src, VString &dst, unsigned const char conv_table[256])` Convert string `src` to string `dst`, according to code page converter `conv_table`.
- * `void SctEbcdicToAscii(const VString &src, VString &dst)`
Convert string `src` from EBCDIC to ASCII and write it to string `dst`.
- * `void SctAsciiToEbcdic(const VString &src, VString &dst)`
Convert string `src` from ASCII to EBCDIC and write it to string `dst`.

3.1.4 File Name conversions

- * `SCT_IS_DIR_SEP(c)`
Return True for character `c` if it is directory name separator (forward or backward slash).
- * `VString SctFileChangeExt(const VString &name, const VString &new_ext)`
In the string `name` change its extension to `new_ext`. Extension is defined as characters from the end of the string till nearest dot. Return the resulting file name.
- * `VString SctFileExt(const VString &name)`
Return file name extension for the given file `name`.
- * `VString SctFileBaseExt(const VString &name, int leave_ext = FALSE)`
Return the base of the file name contained in the string `name`. File name base is file name from which directory name is removed. If `leave_ext` is FALSE, the file name extension is removed too.
- * `VString SctFileDir(const VString &name)`
Return full file name directory for the given file `name`.
- * `VString SctNormalizeFileName(const VString &name)`
Normalize file `name` by removing "xxx/.." fragments that WIN32 does not like. Example: "/aaa/bbb/..ccc" is converted to "/aaa/ccc".

3.1.5 Miscellaneous

- * `VString SctToPrintable(const char *s)`
Convert string `s` to printable form that has no control characters and return the result.
- * `SctErrCode SctUnrollPicStr(const VString &pic_str, VString &out_str, SctExpr *context, VString &err_msg)`
Unroll picture string `pic_str`: substitute `X(n)` by `XXX...XX` (n times). Write the unrolled string to `out_str`.

3.1.6 VStringList: List of Strings

- * `static VStringList null`
Null String List constant.
- * `VStringList()`
Default constructor constructs null list.
- * `VStringList(const VString &s)`

Constructor that constructs list of one string.

```
* VStringList(const VStringList &vsl)
```

Copy constructor.

```
* VStringList &operator = (const VStringList &s)
```

Assignment operator. Assignment function.

```
* void Clear()
```

Delete all elements of this list.

```
* ~VStringList()
```

Destructor.

```
* int IsNull() const
```

Return True, if this is a null list.

```
* int Length() const
```

Return number of elements in this list.

```
* void AddLast(const VString &str)
```

Add string `str` to this list.

```
* void MergeLast(VStringList &vsl)
```

Merge list `vsl` to the end of this list.

```
* void AddFirst(const VString &str)
```

Add string `str` as first element of this list.

```
* void MergeFirst(VStringList &vsl)
```

Merge string `str` as first element of this list.

```
* void InsertAfter(VStringListElt *vsle, const VString &str)
```

Insert string list element with string `str` after the element `vsle`.

```
* friend ostream &operator << (ostream &ost, VStringList &vsl)
```

Print this string list to ostream `ost` in human-readable form.

```
* VStringListElt *first
```

Pointer to the first element of this list.

3.1.7 VStringArray: Array of Strings

```
* VStringArray()
```

Default constructor constructs null list.

```
* VStringArray(const VStringArray &vsl)
```

Copy constructor.

```
* VStringArray &operator = (const VStringArray &s)
```

Assignment operator.

```
* ~VStringArray()
```

Destructor.

```
* int Length() const
```

Return number of elements in this list.

```
* int IsNull(void) const
```

Return True if this array has no elements.

```
* void AddLast(const VString &s)
```

Add string `s` to this array as the new last element.

- * `friend ostream &operator << (ostream &ost, VStringArray &vsl)`
Print this string list to ostream `ost` in human-readable form.
- * `int no`
Number of elements in the array.
- * `int max_no`
Maximum number of elements in the array.
- * `VString *strs`
Pointer to allocated array.

3.1.8 VIntArray: Array of Integers

- * `VIntArray()`
Default constructor constructs null list.
- * `VIntArray(const VIntArray &vsl)`
Copy constructor.
- * `VIntArray &operator = (const VIntArray &s)`
Assignment operator.
- * `~VIntArray()`
Destructor.
- * `int Length() const`
Return number of elements in this list.
- * `friend ostream &operator << (ostream &ost, VIntArray &vsl)`
Print this string list to ostream `ost` in human-readable form.
- * `int no`
Number of elements in the array.
- * `int max_no`
Maximal number of elements in the array.
- * `int32 *ints`
Pointer to allocated array.

3.2 Text Position Classes

These classes are used to represent all kinds of positions in the text files.

3.2.1 Class SctTxtPosn

Class `SctTxtPosn` represents a line & column position in the source file.

The following public members are available:

- `static SctTxtPosn null`
Null position. Has line number and column number set to 0.
- `static SctTxtPosn infity`

Positive Infinity position. Has line number set to 0 and column number set to 1.

- `static VString bad_file_name`
Fictional non-existent file name.
- `SctTxtPosn()`
Default constructor. Construct null position.
- `SctTxtPosn(const VString &i_file_name, uint32 i_line_no, uint16 i_col_no)`
Construct text position with file name `i_file_name`, line number `i_line_no`, and column number `i_col_no`. File name is converted into file index, and if needed, the new name is added to `SctTxtPosn::file_names`.
- `SctTxtPosn(uint16 i_file_ix, uint16 i_col_no)`
Construct text position with file index `i_file_ix`, line number 1, and column number `i_col_no`.
- `SctTxtPosn(const VString &i_file_name)`
Construct text position with file name `i_file_name`, very large (infinite) line number.
- `SctTxtPosn(const SctTxtPosn &pos)`
Copy constructor.
- `~SctTxtPosn()`
Destructor.
- `const VString &FileName() const`
Return file name of this position.
- `int IsNull() const`
Return True if this is a null position. Otherwise return False.
- `int IsInfty() const`
Return True if this is infinitely large position. Otherwise return False.
- `friend SctTxtPosn operator + (SctTxtPosn p1, int c)`
Increase column number in the text position `p1` by `c`.
- `friend SctTxtPosn operator - (SctTxtPosn p1, int c)`
Decrease column number in the text position `p1` by `c`.
- `friend int operator == (const SctTxtPosn &p1, const SctTxtPosn &p2)`
Return True if text position `p1` is equal to text position `p2`.
- `friend int operator != (const SctTxtPosn &p1, const SctTxtPosn &p2)`
Return True if text position `p1` is not equal to text position `p2`.
- `friend int operator < (const SctTxtPosn &p1, const SctTxtPosn &p2)`
Return True if text position `p1` is lexicographically less than text position `p2`. If `p1` and `p2` point to different files, no lexicographical ordering exists.

- friend int operator <= (const SctTxtPosn &p1, const SctTxtPosn &p2)
Return True if text position **p1** is lexicographically less than or equal to text position **p2**. If **p1** and **p2** point to different files, no lexicographical ordering exists.
- friend int operator > (const SctTxtPosn &p1, const SctTxtPosn &p2)
Return True if text position **p1** is lexicographically greater than text position **p2**. If **p1** and **p2** point to different files, no lexicographical ordering exists.
- friend int operator >= (const SctTxtPosn &p1, const SctTxtPosn &p2)
Return True if text position **p1** is lexicographically greater than or equal to text position **p2**. If **p1** and **p2** point to different files, no lexicographical ordering exists.
- friend int SctCmpPosn(const SctTxtPosn &p1, const SctTxtPosn &p2)
Return the result of comparison of text positions **p1** and **p2**:
 - * -100
different files, no ordering can be established.
 - * -1
p1 is less than **p2**.
 - * 0
p1 is equal to **p2**.
 - * 1
p1 is greater than **p2**.
- friend ostream &operator << (ostream &ost, const SctTxtPosn &pos)
Print position **pos** to ostream **ost** in human-readable form.
- VString String() const
Return human-readable string representation of this position.
- void PutTic(ostream &ost) const
Write this position to ostream **ost** in TIC form.
- SctErrCode GetTic(istream &ist, VString &err_msg)
Read this position from istream **ist** in TIC form.
- static SctStrTable file_names
Static table of file names indexed by file indices. Every file index used in every text position indexes a string in this table.
- static int put_string_filenames
Control flag: if True, PutTic writes out whole file names, not file indices.
- SctLineNo line_no
Data member: line number in the file, starts at 1.
- SctColNo col_no
Data member: column number in the line, starts at 0.

- SctFileIx file_ix

Data member: Index of file name in file_names. One of the indexed file names may be just a NULL pointer, which means that this position does not reference any particular file.

3.2.2 Class SctTxtPosnRange

This class represents a begin-end range of text positions.

The following public members are available:

- SctTxtPosnRange()
Construct a null position range.
- SctTxtPosnRange(const SctTxtPosn &i_pos)
Copy constructor.
- SctTxtPosnRange(const SctTxtPosn &i_beg, const SctTxtPosn &i_end)
Construct this position range from start position i_beg and end position i_end.
- int IsNull() const
Return True if this position range is null. Otherwise return False.
- friend int operator == (const SctTxtPosnRange &p1, const SctTxtPosnRange &p2)
Return True if position range p1 is equal to position range p2. Otherwise return False.
- friend int operator != (const SctTxtPosnRange &p1, const SctTxtPosnRange &p2)
Return True if position range p1 is not equal to position range p2. Otherwise return False.
- friend ostream &operator << (ostream &ost, const SctTxtPosnRange &pos)
Print position range pos to ostream ost in human-readable form.
- void PutTic(ostream &ost) const
rite this position range to ostream ost in TIC form.
- SctErrCode GetTic(istream &ist, VString &err_msg)
Read this position range from istream ist in TIC form.
- SctTxtPosn beg
Data member: starting position of this position range.
- SctTxtPosn end
Data member: ending position of this position range.

3.2.3 Class SctTxtOffs

This class represents 24-bit unsigned offset in a text consisting of bytes.

Null (non-existent) offset is represented by number $2^{**24} - 1$.

3.2.4 Class SctTxtOffsRange

This class represents text offset range.

The following public members are available:

- `SctTxtOffsRange()`
Construct null text offset range.
- `SctTxtOffsRange(const SctTxtOffs &i_beg, const SctTxtOffs &i_end)`
Construct this offset range from start offset `i_beg` and end offset `i_end`.
- `int IsNull() const`
Return `True` if this offset range is null. Otherwise return `False`.
- `friend ostream &operator << (ostream &ost, const SctTxtOffsRange &pos)`
Print offset range `pos` to ostream `ost` in human-readable form.
- `unsigned int beg`
Starting offset of this offset range.
- `unsigned int xpos`
Saved position of this node in the line.
- `unsigned int end`
Data member: ending offset of this offset range.
- `unsigned int xstart`
Saved xstart indentation offset of this node.
- `unsigned int beg`
Starting offset of this offset range.
- `unsigned int xpos`
Saved position of this node in the line.
- `unsigned int end`
Data member: ending offset of this offset range.
- `unsigned int xstart`
Saved xstart indentation offset of this node.

3.3 Positioned Token Classes

Token classes used in the parser and on the tree are declared in the file `postoken.h`.

3.3.1 Class SctToken

Class **SctToken** represents a token. The token has a code that identifies it, a string value, and the token knows its position in the original source file.

The following public members are available:

- **SctToken()**
Default constructor.
- **SctToken(int16 i_code)**
Construct a token with code **i_code** and no value.
- **SctToken(int16 i_code, const VString &i_value, const SctTxtPosnRange &i_posn)**
Construct a token with code **i_code**, value **i_value**, and position **i_posn**.
- **SctToken(const SctToken &t)**
Copy constructor.
- **~SctToken()**
Destructor.
- **int IsSeparator()**
Return True if this token is a separator in Cobol sense. Otherwise return False.
- **int IsSpace() const**
Return True if this token is a white space. Otherwise return False.
- **friend int operator == (SctToken &t1, SctToken &t2)**
Return True if token **t1** is equal to token **t2**. Otherwise return False.
- **friend ostream& operator << (ostream &ost, SctToken &t)**
Print token **t** to ostream **ost** in human-readable form.
- **VString String() const**
Return human-readable string representation of this token.
- **VString Image() const**
Return image of this token exactly as it appeared in the original source.
- **void DebugPrint(ostream &ost) const**
Debug-print this token.
- **void PutTic(ostream &ost) const**
Write the TIC form of this token to ostream **ost**.
- **SctErrCode GetTic(istream &ist, VString &err_msg)**
Read this token in TIC form from the istream **ist**. If token cannot be read, write error message to **err_msg** and return error code.
- **void SetCode(int i_code)**
Set this token code to **i_code**.

- `void SetValue(const VString &vs)`
Set value of this token to `vs`.
- `int16 code`
Data member: Token code.
- `VString value`
Data member: Token value.
- `SctTxtPosnRange posn`
Data member: Token position on the source file.

3.3.2 Class `SctTokenListElt`

This is an element of the list of tokens.

The following public members are available:

- `SctTokenListElt(SctToken *i_token, SctTokenListElt *i_next)`
Construct token list element with token `i_token` and next element `i_next`.
- `SctTokenListElt(const SctTokenListElt &)`
Do not copy individual token list elements. This constructor aborts the application.
- `~SctTokenListElt()`
Destructor.
- `SctToken *token`
Data member: pointer to the token sitting in this list element.
- `SctTokenListElt *next`
Pointer to the next element in the list.

3.3.3 Class `SctTokenList`

This is a list of tokens.

The following public members are available:

- `static SctTokenList null`
Empty (null) list constant.
- `SctTokenList()`
Default constructor. Constructs empty list.
- `SctTokenList(const SctTokenList &tl)`
Copy constructor.
- `void Clear()`
Delete all tokens from this list, and their list elements.

- `SctTokenList &operator = (const SctTokenList &tl)`
Assignment operator.
- `~SctTokenList()`
Destructor.
- `int IsNull(void) const`
Return True if this list is empty, False if it is not empty.
- `int Length(void) const`
Return length of this list.
- `void AddLast(SctToken *&t)`
Add token `*t` to the end of this list and fully consume token `*t`. `t` is NULLified after this operation.
- `void MergeLast(SctTokenList &tl)`
Add tokens from list `tl` to the end of this list and fully consume them. `tl` becomes empty after this operation.
- `void MergeLast(SctTailedToken *tt)`
Add tokens from the head and tail of the tailed token `*tt` to the end of this list fully consuming `*tt`. Tailed Token `*tt` is deleted after this operation.
- `void MergeLastTail(SctTailedToken *tt)`
Add tokens from the tail of the tailed token `*tt` to the end of this list fully consuming `*tt`. Tailed Token `*tt` is deleted after this operation.
- `SctToken *FindFirstNonComment()`
Return pointer to the first non-space non-comment token in this list. If no such token is found, return NULL.
- `SctToken *FindLast()`
Return pointer to the last token in this list. If this list is empty, return NULL.
- `void AddFirst(SctToken *&t)`
Add token `*t` to the beginning of this list and fully consume token `*t`. `t` is NULLified after this operation.
- `void MergeFirst(SctTokenList &tl)`
Add tokens from list `tl` to the beginning of this list and fully consume them. `tl` becomes empty after this operation.
- `SctToken *DetachFirst()`
Detach first token from this list and return it.
- `SctToken *DetachLast()`
Detach last token from this list and return it.

- `void DeleteAllThat(int (*criterion)(const SctToken &t))`
Delete all tokens that satisfy `criterion`.
- `void DeleteSpaces()`
Delete white-spaces tokens from this list.
- `friend void SctMergeTokenList(VStringList &sl, SctTokenList &tl)`
Merge strings produced from the token list `tl` to the end of string list `sl`. Fully consume `tl`.
- `void DebugPrint(ostream &ost) const`
Debug-print this token list. Prints one line for every token in this list.
- `friend ostream &operator << (ostream &ost, const SctTokenList &tl)`
Print token list `tl` to ostream `ost` in human-readable form.
- `void PutTic(ostream &ost) const`
Write this token list in TIC form to ostream `ost`.
- `SctErrCode GetTic(istream &ist, VString &err_msg)`
Read this token list in TIC form from the istream `ist`. If it cannot be read, write error message to `err_msg` and return error code.
- `SctTokenListElt *first`
Data member: pointer to the first element of this list.

3.3.4 Miscellaneous functions

These are miscellaneous functions that deal with tokens and token lists.

- `void SctAddMsg(SctTokenList &msgs, int err_type, const SctTxtPosn &posn, const VString &err_text)` Add diagnostic (error) message to the list of messages `msgs`. The list of messages is represented as a list of special message tokens. The added message consists of error type `err_type`, error position in the source `err_posn`, and error message text `err_text`.
- `int SctHasNullPosn(const SctTokenList &tl)`
Return True if token list `tl` contains non-space tokens with NULL position.

3.3.5 Class SctTailedToken

This class represents a token that has head – the "payload" part of the token, and "tail" – comment and whitespace tokens that precede the payload head token. Here tail goes before the dog.

The following public members are available:

- `SctTailedToken(SctToken *i_head, SctTokenList &i_tail)`
Construct tailed token with head `i_head` and tail `i_tail`.
- `SctTailedToken(SctToken *i_head)`
Construct tailed token with head `i_head` and not tail.

- `SctTailedToken(const SctTailedToken &tt)`
Copy constructor.
- `~SctTailedToken()`
Destructor.
- `friend ostream& operator << (ostream &ost, SctTailedToken &tt)`
Print tailed token `tt` in human-readable form to ostream `ost`.
- `void DebugPrint(ostream &ost) const`
Debug-print this token.
- `SctToken *head`
Data member: The pointer to "payload" or head token.
- `SctTokenList tail`
Data member: comment or whitespace tokens that precede the payload token.

Chapter 4

Object Configuration

In large programming projects we are dealing with objects that need to be configurable by user. For instance, in CobolTransformer we have `Parser` object, `PrettyPrinter` object and transformation objects that all have configuration parameters, such as dialect of Cobol to be parsed (in `Parser`) or indentation step (in `PrettyPrinter`).

In this Chapter we propose a template-based mechanism for describing configuration parameters (options) that allows:

- **Command-Line and Visual Configuration**

Configuration parameters, once described in our framework, can be configured by user both in command-line and visual (GUI) environments.

- **Human-Readable Parameter files**

Configuration parameters can be saved to text files in the form `name = value` and loaded from file.

- **Parameter Marshalling**

Configuration parameter blocks can be sent across the network in platform-independent format.

4.1 Option Descriptor

Command-Line/Visual option descriptor (class `SctOptionDesc`) specifies option template for one option variable. `SctOptionDesc` class has the following members:

- `SctOptionDesc()`

Default constructor. Construct an empty option that may be used to end array of options.

- `SctOptionDesc(char i_type, char *i_name, int32 *i_pint_val, VString *i_seln_items, VString *i_pstr_val, int *i_parr_len, int i_max_arr_len, char *i_descr)`

Constructor. Construct an options with specified members.

- `SctOptionDesc(char i_type, char *i_name, int32 *i_pint_val, VString *i_seln_items, VString *i_pstr_val, int *i_parr_len, int i_max_arr_len, char *i_descr, int i_use)`

Constructor. As above, plus specify use flags `i_use`.

- `SctOptionDesc(const SctOptionDesc &od)`

Copy constructor. Usually it is not called, but MS VC++ 6.0 calls it for no reason, so we have it just in case.

- `void PutTic(ostream &ost)`

Output this option in TIC form.

- `char type`

Option type.

* 'b'

Make a Boolean choice: select one of two alternatives: False/True, No/Yes. CL: enter `-name` (True) or `-no-name` (False) for option named `name`. GUI: check box with text `descr`.

Boolean value is stored in `*pint_val`: 0 for False, not 0 for True.

* 'l'

Make a selection: select one of many alternatives. CL: `-name=value1`, `-name=value2`, etc. for option named `name`. GUI: pull-down menu with N items.

N is the number of alternatives (items). Empty `VString()` marks the end of alternatives (items) array. Strings `seln_items[0]` to `seln_items[N-1]` are short names of selection items. Strings `pstr_val[0]` to `pstr_val[N-1]` are long names of selection items. Variable `*pint_val` receives index of the selected item: from 0 to N-1.

* 'n'

Enter decimal integer number. CL: `-name=4`, `-name=-5` or `-name=0` for option named `name`. GUI: enter numeric field, display `descr` text by it. Variable `*pint_val` receives entered 32-bit integer value that can be negative.

* 'a'

Enter array of numbers. Entered array can have 0 or more elements. Numbers in the array can be zero, negative, or positive. CL: `-name=12,24,36` for option named `name`. GUI: Combo Box that enters numbers. Integer `*parr_len` is the length of array of int32 numbers. Integer `max_arr_len` is the maximum length of entered array. Array is not allocated/deleted by us. Array elements `pint_val[0]` to `pint_val[*parr_len-1]` will be populated with entered values.

* 's'

Enter character string. Entered string can be empty. CL: `-name=xxx`, `-name=.` GUI: enter text field, display text `descr` by it. `VString *pstr_val` receives entered string value.

* 'z'

Enter array of character strings. Array can have 0 and more elements. CL: `-name=str1;str2;str3`. Commas or semicolons can be used as separators. GUI: Combo Box that enters strings. Integer `*parr_len` is the length of array of strings. Integer `max_arr_len` is the maximum length of the array. Array is not allocated/deleted by us. `VStrings pstr_val[0]` to `pstr_val[*parr_len-1]` are populated with values of entered strings.

* '-'

Tab(GUI) or Block(CL) separator. String `descr` is Tab/Block header text.

- `char *name`
Option name. A pointer to static string. CL: used as a command line flag name. GUI: not used.
- `int32 *pint_val`
Pointer to integer value(s). Used in 'b', 'l', 'n', and 'a' options.
- `VString *seln_items`
Selection items. Used in 'l' option.
- `VString *pstr_val`
Pointer to string value(s). Used in 'l', 's', and 'z' options.
- `int *parr_len`
Pointer to array length. Used in 'a' and 'z' options.
- `int max_arr_len`
Maximum length of array. Used in 'a' and 'z' options.
- `char *descr`
Option description. A pointer to static string. CL: printed when user requests `-help`. GUI: short text displayed by the option entry field.
- `int use`
Use flags.

4.2 List of Option Descriptors

Class `SctOptionDescList` represents a list of options. This class is what you attach to the object in need of option templates. `SctOptionDescList` has the following members:

- `SctOptionDescList()`
Default constructor. Construct empty list.
- `SctOptionDescList(const SctOptionDescList &)`
- `SctOptionDescList &operator = (const SctOptionDescList &)`
Copy/assign of options list is not possible, because the new option template copy would be pointing to variables in the old option template, while it must point to variables in the new option. So we just create empty lists of options and expect user to call `SetOptions()` on your object when options are needed.
- `void SetTo(const SctOptionDesc &o1)`
- `void SetTo(const SctOptionDesc &o1, const SctOptionDesc &o2)`
- `void SetTo(...)`
Set this option list to a list of specified options: one option, two options, and so on, up to 6 options.

- `void SetTo(SctOptionDesc *opts, int use)`
Set this option list to options from array `*opts`. Option array `*opts` is finished by an empty option. Only options whose use is listed in `use` bit flags are transferred.
- `SctErrCode AddOptions(SctOptionDesc *opts, VString &err_msg, int opt_use = SCOPT_DIR)`
Add option descriptors from array `*opts` that are allowed by `opt_use`. Option array `*opts` is finished by an empty option.
- `SctErrCode AddOptions(const SctOptionDescList &opts, VString &err_msg, int opt_use = SCOPT_DIR)`
Add option descriptors from the list `opts` to this list. Only options that are allowed by `opt_use` are added.
- `SctErrCode AddOption(const SctOptionDesc &opt, VString &err_msg)`
Add to this list option descriptor `opt`.
- `SctErrCode ParseCommandLine(int first_arg, int argc, char *argv[], VStringList &file_name_list, VStringList &msgs)`
Parse command line arguments designated by `first_arg`, `argc` and `argv`. They are a mix of options that start with '-' and several file names (file names do not have '-' in front of them). File names are written to `file_name_list`. Error messages are added to `msgs`. If errors were detected in command line, code `SCT_ERR_BAD_OPTION` is returned. If no errors were detected, `SCT_OK` is returned.

This function does not assign any default values to option variables. If option variable is not changed by the option, then it simply retains its value.

Parse command line arguments from the string `arg_line`. They can be any mix of `-name` and `-name=value` options separated by space. Spaces inside double quotes " " are not considered to be separators. If double quotes are used, the whole option should be enveloped in them. File names are not allowed in `arg_line`, only `-name[=value]` options. If error occurs while parsing command line, add error message to `msgs` and return error code after all options are parsed. This function does NOT assign default values.
- `SctErrCode PrintOptions(ostream &ost, int first_arg, int argc, char *argv[], VString &err_msg)`
Print to ostream `ost` command line arguments designated by `first_arg`, `argc` and `argv` and the current values of the options that were set by option templates in this list.
- `void PutTic(ostream &ost, int use = SCOPT_SAVE)`
Output this option list in TIC form to ostream `ost`. Only output options that are allowed by use flags `use`.
- `void PutTic(VStringList &args, int use)`
Output this list in TIC form to string list `strs`. Only output options that are allowed by use flags `use`.
- `SctErrCode GetTic(istream &ist, SctTokenList &msgs)`
Get this option descriptor list in TIC form from istream `ist`. Concatenation of error messages is written to `msgs`. The whole file is parsed even if errors were detected. If `SCT_OK` is returned, no errors were detected in TIC file.

- `int descrs_no`
- `SctOptionDesc descrs[SCT_MAX_CL_OPTIONS]`
Data members: actual array of option descriptors.

Chapter 5

Parsing

5.1 SctParser class

To parse Cobol programs using CobolTransformer you need to have an object of class derived from **SctParser** declared in your program. **SctParser** class contains function **Parse** that is called to parse a Cobol source file. Also **SctParser** class contains a number of configuration parameters that affect the behaviour of the parser. Once parsed, Cobol program is returned to you in the form of the Program Tree.

SctParser class is a pure virtual class that serves an interface to the parser. The actual parser implementation class is inherited from **SctParser** and it can be either **SctParserExcl** from **parser-excl.h** for client-server setup (Executable License) or **SctParserImpl** from **parser-impl.h** for the direct link setup (Library License).

Currently you can have at most one object of class **SctParser** in your program. This is because the parser generator that we use (BtYacc) employs static variables. This restriction will be removed in one of the future releases.

The **SctParser** class is declared in the file **parser.h**. The class has the following public members:

- **SctParser()**

Default constructor. Please note that no explicit copy constructor and assignment operator are declared, because we use implicit copy constructor and assignment operator.

- **~SctParser()**

Destructor.

- **SctErrCode Parse(const VString &file_name, SctExpr *&prog, SctTokenList &err_msgs)**

Parse Cobol program. Input parameters:

- * **file_name**

- Name of the main file to parse.

- * **prog**

- Parser writes the pointer to the resulting tree to this reference. You own the returned tree and you are responsible for deleting it.

- * **err_msgs**

- Parser writes recoverable error and warning messages to **err_msgs**.

You may need to set configuration parameters described below. They all have reasonable default values but these values are not necessarily what you need.

Return code:

- * **SCT_OK**

- * **SCT_ERR_PARSER_ERRORS**

Program was parsed till the end, and the program tree is deposited into ***prog**. However, errors and warnings could have occurred while parsing. Error and warning message are written to **err_msgs**.

- * **SCT_ERR_PARSER_SEVERE**

Parser could not finish parsing the program. The code of the error that caused the problem is returned. Error message for this error is the last message at **err_msgs**. Typically this happens when parser cannot recover from severe syntax error.

If really bad Severe or Terminal internal error occurs, then ***PError** function is called. The only correct course of action for ***PError** is to terminate the whole application.

- **void (*pPError)(int err_type, const SctTxtPosn &pos, const VString &s)**

Severe/Terminal error processing function supplied by the user. It must inform the user of the error and terminate application. Parser cannot function normally after severe/terminal error has occurred. These are the system errors that do not occur under normal circumstances. It is unlikely that this function will be called.

- **SctOptionDescList options**

Object configurator for the parser. This list of option templates is used to set configuration parameters of the parser from command line or GUI environment. Do not use **options** if you set configuration parameters only programmatically.

- **void SetOptions(int use = SCOPT_ALL)**

Prepare option templates for work. This function must be called before you use option templates for setting configuration parameters from command line or GUI. Do not call **SetOptions** if you set configuration parameters only programmatically.

- **SctErrCode SetConfig(VString &err_msg)**

Check correctness and consistency of all configuration parameters. Compute derived parameters and make settings effective. If configuration parameters are inconsistent, error message is written to **err_msg** and error code is returned.

- **SctCobolDialects cbl_dialects**

A set of Cobol dialects to be recognized by this parser (configuration parameter). Every individual field in this object is responsible for one dialect. An arbitrary union of dialects may be recognized by the parser.

Do not set **cbl_dialects** if you call **SetConfig()**, because **SetConfig()** itself computes **cbl_dialects** based on **cbl_dialect_1** and **cbl_dialect_2** below.

- **int cbl_dialect_1**

Primary Cobol dialect index (configuration parameter). Can be any of the values declared in **enum SctCobolDialectIx** from **dialects.h**. When you call **SetConfig()**, this primary dialect index is used to compute **cbl_dialects** dialect set.

Default: `COBOL_MF`.

- `int cbl_dialect_2`

Secondary Cobol dialect (configuration parameter). Can be any of the values declared in `enum SctCobolDialectIx`. When you call `SetConfig()`, this secondary dialect index is used to compute `cbl_dialects` dialect set.

Default: `COBOL_NONE`.

- `SctCobolDialects allowed_dialects`

Set of allowed dialects, used in computing `cbl_dialects` by `SetConfig()`. If `cbl_dialect_1` or `cbl_dialect_2` specify dialect that is not allowed by `allowed_dialects`, then `SetConfig()` returns error that says that un-allowed dialect was used.

Default: all dialects are allowed.

- `int line_format`

Source line format (configuration parameter):

- * `SCT_SLF_DIAL`
determined by the dialect,
- * `SCT_SLF_FIXED`
fixed source line format,
- * `SCT_SLF_FREE`
free line format,
- * `SCT_SLF_FSC_FREE`
Fujitsu Cobol free line format.
- * `SCT_SLF_VAR`
variable line format.

Default: determined by the dialect. For FSC Cobol default line format is `SCT_SLF_VAR`, for ICobol it is `SCT_SLF_FREE` and for all other Cobol dialects default line format is `SCT_SLF_FIXED`.

- `int progid_comments`

If `TRUE`: Allow comments to immediately follow `PROGRAM-ID` paragraph on the same line (configuration parameter).

Default: `FALSE`.

- `int sep_space_reqd`

Require space after separators (configuration parameter).

Possible values: 0 - determined by the dialect, 1 - do not require space, 2 - require space after separators.

Default: 0, determined by the dialect. MF, MS and SAA dialects do not require space, all others do require space.

- `VStringArray excluded_keywords`

Excluded keywords (configuration parameter). This is array of `VStrings` that contain Cobol words that must be treated as user-defined names by the parser, not as keywords.

Default: no excluded keywords.

- **VStringArray set_constants**

Provides the same function as the Micro Focus CONSTANT compiler directive. This configuration parameter allows you to use constant names in Cobol program but set their values on the command line. The effect is as if you declared a 78-level item with that value.

Each element in this array of strings defines one 78-level constant.

Format of the definition for string value is:

const-name 'string-value'

Format of the definition for numeric value is:

const-name(numeric-value)

Quotes indicate that the value is a string; parentheses indicate that it is a numeric value. You must use single quotes, not double quotes, and there must be no spaces between any items in the string.

Default: no external constants defined.

- **int assign_external**

If **TRUE**: If neither **EXTERNAL** nor **DYNAMIC** are specified in the **ASSIGN TO** clause, tells the parser to assume the (Micro Focus) assignment is **EXTERNAL** (configuration parameter).

If **FALSE**: assume the assignment to be **DYNAMIC**.

Default: **FALSE**.

- **int sql**

Parse SQL in **EXEC SQL** statements (configuration parameter). If this option is **FALSE**, **EXEC SQL** statements are passed thru as a sequence of tokens. If this option is **TRUE**, **EXEC SQL** statements are parsed into a tree, so that they can be analyzed and transformed.

Default: **TRUE**.

- **int cics**

On: Parse CICS statements embedded into **EXEC CICS ... END-EXEC**. Off: let CICS go through as a sequence of tokens.

Default: **FALSE**.

- **int cics_eib**

Add CICS EIB data block at the beginning of **LINKAGE SECTION**.

Default: **FALSE**.

- **VStringArray copylib_dirs**

List of directories searched for **COPY** library files (configuration parameter). This is array of **VStrings** that each contain a directory name.

Directory names can be absolute or relative to the working directory.

Both **"/** (slash) and **"** (backward slash) can be used as separators of individual directory names in full directory names.

Default: only the current working directory (**"/.**) is searched.

- **VString copylib_sfx**

Configuration parameter that defines the extension (suffix) to append to COPY library file names when the extension is not specified in the program source (as in COPY file).

When file name suffix is explicitly specified, as in COPY "file.ext" then the default suffix is not added.

Note that the period (".") needs to be specified as part of copylib_sfx.

If copynames_case is SCT_COPYCASE_LOWER and COPY file "FiLe.SFX" is not found, try lowercase version "file.sfx" of the file name.

If copynames_case is SCT_COPYCASE_UPPER and COPY file "FiLe.sfx" is not found, try uppercase version "FILE.SFX" of the file name.

If copynames_case is SCT_COPYCASE_EXACT then COPY file name should match exactly the actual file name.

Please note that on WIN32 platforms file name capitalization is ignored by the system, so copynames_case=SCT_COPYCASE_EXACT will work just fine. This means that using this options on WIN32 will not change anything.

Debugging this: if you want to monitor how parser tries to locate copybooks and other file-related activities, set environment variable SCT_DEBUG to 1.

Default: ".cpy"

- **int copynames_case**

If a COPY file cannot be located, gives the character case the tool should use for the file name in further open attempts (configuration parameter):

- * SCT_COPYCASE_LOWER
try lower case,
- * SCT_COPYCASE_UPPER
try upper case,
- * SCT_COPYCASE_EXACT
do not try upper/lower case, use the exact name.

Default: SCT_COPYCASE_LOWER.

- **int old_copy**

If TRUE: replace data item declaration with the text from copylib that is COPYd in this data item declaration (configuration parameter). This option is used to emulate behaviour of ANSI-1968-based compilers. Equivalent to OLDCOPY directive of Micro Focus compiler.

Default: FALSE.

- **int irrev_inline**

When the irrev_inline configuration parameter is FALSE the lexical analyzer inserts SEP_INLINED_COPY and SEP_COPY_EOF tokens in the token stream indicating the beginning and end of copybook code. With these tokens, the parser can place STMT_INLINED_COPY_BEG and STMT_INLINED_COPY_END statements on a Program Tree and therefore it can output separate copybook files or place enter and exit copybook comments in the Cobol file (controlled by the gen_copy_dir and gen_enter_exit_copy_comments options).

However, certain Cobol structures involving copybooks cannot be parsed with the `SEP_INLINED_COPY` and `SEP_COPY_EOF` tokens in the token stream, because in general case, `SEP_INLINED_COPY` and `SEP_COPY_EOF` tokens are not part of the Cobol grammar. To parse these structures, or if you always want the copybook code expanded inline, set the `irrev_inline` option to `TRUE`.

Default: `FALSE`.

- `int inline_copy`

If `TRUE`: parse copybook files referred by the `COPY`, `--INC` and `INCLUDE++` statements (configuration parameter). If `FALSE`, do not go into the copybooks as if they were not found.

Default: `TRUE`. We do not recommend to turn this option off.

- `int display_warnings`

If `TRUE`: add warnings messages to the message list. If `FALSE`: do not add warnings to the message list.

Default: `TRUE`.

- `int multiple_undef_errs`

If `TRUE`: issue error message once per every use of the undefined data-item. If `FALSE`: only one error message is issued on the first use of the undefined name.

Default: `FALSE`.

- `int resolve_use_def_links`

If `TRUE`: resolve definition-use links for all named objects. If `FALSE`: do not resolve def-use links – this might save some time when parsing.

Default: `TRUE`.

- `int same_proc_data_name`

Allow same user-defined word to be used both as data-name and paragraph-name. Possible values: 0 - determined by the dialect, 1 - no, 2 - yes.

Default: 0, determined by the dialect: Yes for MF and ICobol, No for other dialects.

- `int compute_leng_offs`

If `TRUE`: Compute data items length and offsets. Not computing lengths and offsets saves some time, but makes some transformations impossible.

Default: `TRUE`.

- `int lo_stor_mode`

Type of alignment (storage mode) used to compute length and offset of data items. If it is `SCT_STOR_MODE_BYTE`, the binary items length can be any number of bytes from 1 to 8 and they can be aligned on the byte boundary. If it is `SCT_STOR_MODE_WORD`, the binary items length can be 2, 4, or 8 bytes and they are aligned on 2, 4, or 8-byte boundary.

If storage mode is `SCT_STOR_MODE_DIAL`, then the actual storage mode is determined by the dialect. Default mode is `SCT_STOR_MODE_DIAL`.

- **int lo_sign_trailing_separate**
Flag: Is SIGN for numeric items a trailing separate character. Values: 0 – determined by the dialect, 1 – No, 2 – Yes.
Default: 0.
- **int lo_pointer_size**
Size of the POINTER data item. Depends on particular cobol system/CPU architecture.
Default: 4.
- **int lo_proc_pointer_size**
Size of the PROCEDURE-POINTER data item. Depends on particular cobol system/CPU architecture.
Default: 4.
- **int lo_index_size**
Size of the INDEX data item. Usually it is equal to BINARY or POINTER size.
Default: 4.
- **int lo_unfold_flex_arrays**
How to treat DEPENDING ON clauses. If **unfold_flex_arrays** is On, then size of table with DEPENDING ON is computed as difference between upper and lower bound on indices. If this is Off, then size table with DEPENDING ON is undefined.
Default: TRUE.
- **int null_nodes**
If TRUE: Apply NullPtrsToNullNodes transformation to Program Tree after parsing to make sure that nodes never refer to NULL pointers. As a result, in the Program Tree all pointers to the children are non-NULL. If this flags is FALSE, then pointers to some non-existing children will be NULL.
Default: TRUE.
- **int preserve_exec_format**
If TRUE: preserve EXEC statement internals formatting.
Default: TRUE.
- **void (*p_ProcessExpr)(SctExpr *&e)**
Callback function: user-supplied Expression processor. Not available in Client-Server mode.
At certain Program Tree nodes that represent sizable chunks of Cobol Program such as small Divisions, Sections, Data Item Definitions parser can call a user-defined function provided here.
Usually you do not need to define this function, because you will get the whole Program Tree after parsing is done.
However, if you are concerned about Program Tree taking too much memory, you can define this function, so that you can process and free chunks of Program, before the whole Program takes too much memory.
Default: no expression processor, build the whole tree.

- `void (*pProgress)(SctTxtPosn &pos)`
 Callback function: user-supplied Progress Indicator display function. This function receives current parsed text position and commenting text that may be empty. Not available in client-server mode.
 Default: no progress indicator.
- `int progress_step`
 Control parameter: How many lines are processed before `*pProgress` is called. Not available in client-server mode.
 Default: 100.
- `int lex_only`
 If `TRUE`: do lexing only and no parsing. For testing only. Not available in client-server mode.
 Default: `FALSE`.
- `int debug_lex`
 Control parameter: lexer debugging bits (bitfield).
 Default: no debugging.
- `int debug_grammar`
 Control parameter: BtYacc grammar debugging level.
 Default: no debugging.
- `int debug_symbol`
 Control parameter: symbol table debugging level.
 Default: no debugging.
- `int debug_posn`
 Control parameter: position assignment debugging level.
 Default: no debugging.
- `int debug_tic`
 Control parameter: output TIC debugging level.
 Default: no debugging.
- `int print_parsed_tree`
 Control flag: Print the Program Tree that is result of parsing.
 Default: no printing.
- `int32 ctr_total_tokens`
 Counter: number of tokens in the parsed source counted at the `GetToken()` level. Written by the parser.
- `int32 ctr_total_lines`
 Counter: total lines in the parsed source counted at the `GetLine()` level. Written by the parser.

- `int32 ctr_lines_of_code`

Counter: total lines of code in the parsed source counted at the `GetLine()` level. Written by the parser.

How to invoke parser. You need to perform the following steps to invoke parser:

- Create `SctParser` object

Declare object of class `SctParser` in your program. Default constructor is run that sets configuration parameters to their default values.

- Set Configuration Parameters

If default values set in `SctParser` do not satisfy you, set the parser configuration parameters described above.

If you use `SctOptionDesc` object configurator, call `SetOptions` and then call either `ParseCommandLine` or other option configuration routine. There is no need to call `SetOptions` if you do not use `SctOptionDesc` object configurator.

- Call `SetConfig()`

Call `SctParser::SetConfig()` function that makes sure that configuration parameters are correct and computes derived configuration parameters.

- Call `Parse()`

Now that your configuration parameters are set and verified, call `Parse()` to perform the actual parsing.

- Display error messages

Parser may return error messages that say that parser cannot understand certain parts of your Cobol program. Typically you want to display the error messages returned by the parser.

- Decide whether to proceed

Even if errors were detected, parser is able to build Program Tree most of the time and this tree is returned to the program. If severe errors occur, no tree is built. The easiest way to check that tree is present is to check that the variable `prog` assigned to by the parser is not `NULL`. You always can try to convert an incomplete tree returned by partially erroneous parse of the Cobol source.

5.1.1 Parser Features

Dialects. The Cobol grammar is written in BtYacc (Backtracking Yacc). The grammar is a union of several popular Cobol dialects.

The actual job of distinguishing between these dialects is mostly left to the lexer. In the lexer the Cobol dialect set by the user affects determination of whether Cobol word taken from Cobol program is reserved word (keyword) or user-defined word.

The dialect also affects other aspects of lexer and parser behavior, such as how comments are recognized.

Several dialects that conflict a lot with popular dialects are separated off in separate modules. These modules are licensed separately, and they augment the base grammar – one at a time.

Currently we have separate grammar modules for: (1) Wang Cobol, (2) DMS-80/1100 DML embedded database language, (3) Fujitsu NetCobol.

The dialect-related things are defined in `dialects.h`:

Structure `SctCobolDialects` defines a set of dialects. Since we want parser to be able to accept a union of several Cobol dialects, the dialect specifier is a set.

The currently available Cobol dialects are:

- * **extra**
Extra (temporary) keyword. Used in reserved words table only.
- * **ansi74**
ANSI-74 Cobol standard.
- * **ansi85**
ANSI-85 Cobol standard.
- * **osvs**
IBM OSVS Cobol.
- * **vsii**
IBM VS COBOL II revisions 3, 4 (ANSI-85). IBM VS COBOL II revisions is union `osvs|vsii`.
- * **saa**
IBM SAA Cobol definition levels 1, 2.
- * **xopen**
X/Open Cobol definition (ANSI-85).
- * **mf**
Micro Focus Cobol (ANSI-85).
- * **ms**
Microsoft Cobol.
- * **rm**
Ryan McFarland Cobol (ANSI-74).
- * **rm85**
Ryan McFarland Cobol-85 (ANSI-85).
- * **dosvs**
IBM DOSVS Cobol.
- * **univac**
UNIVAC Cobol (partially implemented).
- * **wang**
Wang VS Cobol (ANSI-85).
- * **fsc**
Fujitsu Cobol-85 (ANSI-85).
- * **net**
Fujitsu/Synkronix NetCobol (ANSI-85 based).
- * **icobol**
ICobol.
- * **dml**
Embedded language DMS/80 and DMS/1100 DML (CODASYL for UNISYS mainframes).

Must be used with one of the Cobol dialects.

* **idms**

Embedded language CA-IDMS DML (CODASYL on IBM mainframes, not implemented).
Must be used with one of the Cobol dialects.

These are the functions for dialect set handling:

- **SctCobolDialects DialectsNone()**

Returns empty dialect set.

- **SctCobolDialects DialectsAll()**

Returns full dialect set, in which all dialects are present.

- **VString SctDialToString(SctCobolDialects &dials)**

Convert dialects set to the string that encodes the set and return the encoding string.

- **SctErrCode SctStringToDials(SctCobolDialects &dials, const VString &s, VString &err_msg)**

Convert dialect set encoding string **s** to the dialects set **dials**.

- **void SctSetDialect(SctCobolDialects &dial, SctCobolDialectIx dial_ix)**

Set dialects set **dial** according from the numeric dialect index **dial_ix**.

- **SctErrCode SctSetDialects2(SctCobolDialects &dials,
SctCobolDialectIx dial1_ix, SctCobolDialectIx dial2_ix,
SctCobolDialects &allowed, VString &err_msg)**

Set dialect set **dials** to a union of two dialects represented by their numeric indices **dial1_ix** and **dial2_ix** and check that these dialects are among **allowed** dialects. Return error in **err_msg** if the newly combined dialect is not allowed.

- **int SctIsDialectSet(SctCobolDialects &dials, SctCobolDialectIx dial_ix)**

Return True if dialect **dial_ix** is set in dialect structure **dials**. Otherwise return False.

- **VString SctDialToPrintableString(SctCobolDialects &dials)**

Convert dialects bitfield to printable string.

The Cobol dialect affects the following:

- * How reserved words are recognized. Each dialect has its own list of reserved words as specified in the file **keywords.cxx**.
- * For MicroFocus allow separators not be finished with space.
For instance, **A(2)=B** would be legal, while in other Cobols you must write **A(2) = B**.
- * For MicroFocus allow “*>” in-line comments.
- * For Wang allow “%” in-line comments.
- * For MicroFocus allow **\$SET**, **\$IF**, **\$ELSE** and other **\$** directives.
- * For OSVS allow reserved words **PROGRAM-ID**, **ENVIRONMENT**, **DATA**, **PROCEDURE** to finish the **REMARKS** paragraph even if they appear in Zone B.
In all other dialects the finishing reserved word should appear in Zone A.

Grammar Conflicts and Backtracking. We decided to write the Cobol grammar in Yacc, but Cobol grammar cannot be fully described with LALR(1) grammar used by plain vanilla Yacc.

To address this problem, we use *Backtracking Yacc*, or *BtYacc*. This is Berkeley Yacc plus backtracking feature added by Chris Dodd chrisd@collins.com. Siber Systems have modified BtYacc in several ways to make it suitable for production environment and to fix bugs. The modified BtYacc is available from <http://www.siber.com/btyacc/>.

If there is a shift-reduce conflict in the grammar, BtYacc installs a checkpoint, goes into trial mode and tries shift first. Then if grammar rule states (using special YYACCEPT action) that the chosen path is a correct one, parser rolls back to the checkpoint and reparses saved tokens executing all the semantic rules in full.

However, if the chosen path results in Yacc error (implicit or explicit), then parser rolls back to the checkpoint and tries the reduce path.

This approach allows us to correctly parse Cobol constructs that require lookahead of more than 1. It also makes semantic feedback to the parser easy to implement.

The current list of official conflicts is contained at the beginning of the grammar file `cobol.y`.

One Source File – One Program Tree. The whole source Cobol file is parsed into one Program Tree.

However, if in your particular application Program Trees are expected to become big, then you can define your own user-defined `ProcessExpr()` function that will process a small subtree of a program.

After completing processing of the tree fragment, `ProcessExpr` function may swap the processed tree out to the disk file for further processing, or just delete the tree if the performed processing was final.

Our biggest Program Tree took only 25Mb of PC memory which is not too big even by consumer PC standards.

Inlined COPY and -INC statements. The SourcePrint capability allows user to exactly restore the original structure of copybooks or to completely change the layout of copybooks.

To make this possible we had to introduce nodes that represent copybook start and finish on the Program Tree. By changing and moving these special nodes you can try exercise a fine control over generation of copybook files in SourcePrint.

These nodes are:

- **STMT.INLINED_COPY_BEG**

This node marks a start of a copybook.

Hidden argument \$3 (`$AX_COPY_REAL_FILE`) contains the real file name as determined at parsing time.

Hidden argument \$4 (`$AX_COPY_LEVEL`) contains the nesting level of this copybook start. The nesting level must be consistent with the nesting level variable that SourcePrint maintains when printing the Program Tree out.

When this node is source-printed in copybook creation mode, the COPY statement is printed to the current output file and the copybook output file is open, the current output file is saved on the stack and the copybook is made a current output file.

- **STMT_INLINED_COPY_END**

This node marks an end of copybook.

Hidden argument \$1 (\$AX_EOF_REAL_FILE) contains the real file name as determined at parsing time.

Hidden argument \$2 (\$AX_EOF_LEVEL) contains the nesting level of this copybook end.

There must exist matching **STMT_INLINED_COPY_BEG** node that lexicographically precedes this node.

When this node is source-printed in copybook creation mode, the current output file is closed, and we return to the output file stored in the stack of output files.

Uninlined COPY statements. Sometimes inlining of a COPY statement does not happen. There may be several reasons for this:

- The file specified in COPY statement is not available.
- Flag `lexer.inline_copy` is set to FALSE by user, he does not want inlining to happen.

When we cannot inline the COPY library file, a smooth parsing process is broken, because parser does not know what is inside the COPY library file.

However, we still want the parser to continue parsing and recover gracefully from this situation, because:

- We still want the COPY statements to appear in the resulting program.
- We do not want the partially parsed syntactic unit that precedes COPY statement be lost. We want this unit to make it to the output program.

To achieve this, our Lexer generates **SEP_UNINLINED_COPY** token for each uninlined COPY statement.

Parser accommodates **SEP_UNINLINED_COPY** as a legitimate token whenever possible and puts it on the Program Tree either as a Statement Node or as Expression Node depending on context.

At the end, you can have special **EX_UNINLINED_COPY** and **STMT_UNINLINED_COPY** nodes in the Program Tree. These special operations signify the fact that contents of this fragment of the tree are unknown to us, because it is contained within uninlined COPY statement.

Exec Statement. Contents of the EXEC (EXECUTE) statement that appear between EXEC and END-EXEC tokens are not written in Cobol, they are written in a language that may be very different from Cobol both lexically and syntactically.

Therefore contents of EXEC statement require a special parser (and sometimes lexer too). These extra parsers for languages like CICS, SQL are currently outside of the CobolTransformer scope, but they may be supplied as add-on parsers.

Since we cannot parse these languages, we just gather all the tokens that appear between EXEC and END-EXEC and put them on the tree under the EXEC statement, so that other programs can parse and analyze them in any way they like.

If boolean lexer flag `preserve_exec_format` is set, then the original formatting of the EXEC statement contents is preserved, that is, spaces are not compressed and newline characters are not thrown out.

Listing Control Statements. These statements (EJECT, SKIP1, SKIP2, SKIP3, TITLE) only control the way program listing looks and they have no any semantical effect on a program.

So we convert them to comments.

Identification Division Paragraphs. Identification Division paragraphs AUTHOR, DATE-COMPILED, DATE-WRITTEN, INSTALLATION, REMARKS, SECURITY are converted to comments, because comments is what they really are.

5.1.2 Lexer Features

Allow missing spaces between operations and names. In standard Cobol operations +, -, *, /, >, =, < are just a special case of Cobol words, that need to be separated by separators. So in standard Cobol `A=B+C` would be illegal.

In MF Cobol this construct is legal.

When MF dialect is set, we allow these constructs too.

Complete implementation of COPY and REPLACE statements. We have complete, 100%-standard compliant no-questions-asked implementation of COPY and REPLACE statements.

Multiple levels of inclusion are allowed.

Word-part substitution does work.

-INC statement implemented.

REMARKS paragraphs and the likes. Paragraphs that consist entirely of comment entries are converted to comments.

Chapter 6

Cobol Code Generation

PrettyPrint and *SourcePrint* functionality is implemented in the `SctPrettyPrinter` class. This class is declared at the file `pprinter.h`.

6.1 SctPrettyPrinter class

To generate Cobol programs from the Program Tree you need to have an object of class derived from `SctPrettyPrinter` declared in your program. `SctPrettyPrinter` class contains function `PrettyPrintFile` that is called to perform the pretty-printing. Also `SctPrettyPrinter` class contains a number of configuration parameters that affect the behaviour of the pretty-printer.

`SctPrettyPrinter` class is a pure virtual class that serves as an interface to the pretty-printer. The actual parser implementation class is inherited from `SctPrettyPrinter` and it can be either `SctPrettyPrinterExcl` from `pprinter-excl.h` for client-server setup (Executable License) or `SctPrettyPrinterImpl` from `pprinter-impl.h` for the direct link setup (Library License).

You can have several objects of class `SctPrettyPrinter` in your program.

The `SctPrettyPrinter` class is declared in the file `pprinter.h`. The `SctPrettyPrinter` class contains the following members:

- `SctPrettyPrinter()`
Default constructor. Please note that no explicit copy constructor and assignment operator are declared, because we use implicit copy constructor and assignment operator.
- `~SctPrettyPrinter()`
Destructor.
- `SctErrCode PrettyPrintFile(const VString &out_file_name, SctExpr *prog, SctTokenList &err_msgs)`
- `SctErrCode PrettyPrintString(VString &out_str, SctExpr *prog, SctTokenList &err_msgs)`
PrettyPrint Program Tree (expression) `prog` to:
 - * `PrettyPrintFile`
the file named `out_file_name`,
 - * `PrettyPrintString`
the string `out_str`.

Printing is controlled by Code Generation Table that is computed based on the SCT-supplied Default Code Generation table and user-supplied Primary Code Generation table.

SourcePrint is invoked when `gen_src` flag is set to `TRUE`.

PrettyPrint() functions do not write a binary 0 at the end of the stream ost, because we do not want 0 in file streams. If you are pretty-printing to a string stream `strstream` and you want to use the resulting string as C string, then do not forget to append `ends` to `out_str` after pretty-printing. Also we do not close or flush the stream ost.

Return:

- * `SCT_OK`
PrettyPrint() was able to print all of the specified tree. However, errors and messages could have occurred. See error and warnings messages in `err_msgs`.
- * `SCT_ERR_PRINT...`
PrettyPrint() was NOT able to print the specified tree till the end. Error message for the error that caused termination is the last mmessage at `err_msgs`. The last error message says what caused this bailout error. Error code of this error is returned.

If really bad Severe or Terminal internal error occurs, then function `*PError` is called. The only correct course of action in `*PError` is to terminate the whole application.

- SctOptionDescList options

Object configurator for the pretty-printer. This list of option templates is used to set configuration parameters of the pretty-printer from command line or GUI environment. Do not use `options` if you set configuration parameters only programmatically.

- void SetOptions(int use = `SCOPT_ALL`)

Prepare option templates for work. This function must be called before you use option templates for setting configuration parameters from command line or GUI. Do not call `SetOptions` if you set configuration parameters only programmatically.

- SctErrCode SetConfig(VString &err_msg)

Check correctness and consistency of all configuration parameters. Compute derived parameters and make settings effective. If configuration parameters are inconsistent, error message is written to `err_msg` and error code is returned.

- SctGenTblElt *gen_table_primary

Configuration parameter: pointer to user-defined Primary Code Generation Table.

Entries in this table override entries in the Default Code Generation Table. So if you need to modify default code generation table, you do not need to copy the whole Default Table and then change a few entries. Just define the entries you want to change in `gen_table_primary` and will merge these entries into the combined table.

Last entry in the table must be `SctGenTblElt()`. It marks the end of the table and it must be present.

Default: `NULL`, no Primary Code Generation Table defined.

- int gen_set_gen_posn

Configuration flag: if **TRUE**, then for every node **e** of this tree begin and end positions of the **e**'s image in the printed text are written to **e.gen_posn**. Not available in client-server.

Default: **FALSE**.

- **int gen_src**

Configuration flag: if **TRUE**, generate output in **SourcePrint** mode. If **FALSE**, generate output in the standard fully beautified mode.

In the **SourcePrint** mode the output is as close to the original source as possible. Only the newly generated tree nodes that do not originate from the source are beautified.

Default: **FALSE**.

- **VString gen_copy_dir**

Configuration parameter: indicates that copy files should be written to separate files, rather than being expanded (inlined) in the main Cobol file, and defines the directory to which the output copy files should be written.

If set to a null string, the default, copy files are expanded inline in the output file. Only valid if the **SctParser::irrev_inline** was **TRUE** when parsing.

Copybook name that is stored in the Program Tree is appended to the directory name specified in **gen_copy_dir** option to form full name of output copybook file. However, if copybook name stored on the tree is absolute name (this happens if you specify absolute names in **SctParser::copylib_dirs** option), then only base file name derived from this absolute copybook name is appended to output copybook directory name.

If you use **".."** (parent directory) in directory name specified in **SctParser::copylib_dirs**, then substrings **xxx/..** where **xxx** is arbitrary directory will be removed from the resulting copybook name, because WIN32 systems do not accept **".."** in file name.

Default: null string.

- **int gen_indent_step**

Configuration parameter: step of indentation for the beautified code. When the pretty-printer outputs pretty-printed code it uses indentation to help the interpretation of statements that are spread over multiple lines or have nested levels of logic (such as nested **IF** statements). The **gen_indent_step** parameter specifies the number of character positions by which each successive indent is displaced from the previous indent.

Affects handling of **\t** and **\b** special characters.

Default: 2.

- **int gen_max_indent**

Configuration parameter: Defines the maximum column at which indented text will be started. Any text whose indent would start at a column greater than this column is output with the indent starting at this column.

Affects handling of **\t** and **\b** special characters.

Default: 40.

- **VIntArray gen_tabs**

Configuration parameter array: tabulation stops.

When the pretty-printer outputs pretty-printed code certain items, such as PICTURE clauses, are aligned to tab columns. The `gen_tabs` parameter specifies the tab column positions.

For every `i` from 1 to `gen_tabs.no` the table element `gen_tabs.ints[i-1]` is the tabulation position $\$pi$.

Currently tabs are used for data items and report section items:

| Data items | Tab |
|------------|-----|
| PICTURE | \#4 |
| REDEFINES | \#4 |
| USAGE | \#6 |
| OCCURS | \#4 |
| VALUE | \#6 |

| Report section items | Tab |
|----------------------|-----|
| PICTURE | \#3 |
| SOURCE | \#5 |
| VALUE | \#5 |

Default: 6 tabs at columns 12, 24, 32, 42, 44, 54.

- `int gen_line_format`

Configuration parameter: generated line format.

- * `SCT_SLF_DIAL`
determined by the dialect,
- * `SCT_SLF_FIXED`
fixed source line format,
- * `SCT_SLF_FREE`
free line format,
- * `SCT_SLF_FSC_FREE`
Fujitsu Cobol free line format.
- * `SCT_SLF_VAR`
variable line format.

Default: `SCT_SLF_FIXED`.

- `int gen_observe_ab_rules`

Configuration flag: if `FALSE`, do not observe Area A/B rules when generating the code. Ignored if `gen_line_format` is `SCT_SLF_FREE`.

Affects handling of `\t` and `\b` special characters.

Default: `TRUE`.

- `int gen_print_comments`

Configuration flag: If `TRUE`, print comments, when in `!gen_src` mode. If `FALSE`, do not print comments.

Default: `TRUE`.

- `int gen_enter_exit_copy_comments`

Configuration flag: add `== Enter/Exit copy-file.ext ==` comments at the start and end of the inlined copybooks. Ignored if `SctParser::irrev_inline` is `TRUE`.

Default: `TRUE`.

- `int gen_add_line_dirs`

Configuration flag: add FSC preprocessor `\#LINE/\#FILE` directives to the pretty-printed output. Should be used only in `FSC_PP` mode when `gen_src` is `FALSE`.

Default: `FALSE`.

- `int gen_line_id_comments`

Configuration flag: if `TRUE`, transfer characters held in columns 1-6 and 73-80 from the input source to the output source. Effective only when `gen_src` is `TRUE`.

Default: `TRUE`.

- `VString gen_col73to80_fmt`

Configuration parameter: Specify a string to be placed in columns 73 to 80. You can include the line number in the string by including one of the following `"%"` strings (a la C `printf`):

- * `%d`
output the line number using the minimum number of digits (i.e. one digit for the numbers 1-9, two digits for the numbers 10 to 99, etc.),
- * `%nd`
use `n` digits for the line number (right aligned, space filled),
- * `%-nd`
left align the line number in `n` digits,
- * `%0nd`
zero fill a right aligned, `n` digit line number.

You can also substitute the letters `"o"` or `"x"` for the letter `"d"` in the above strings to output the numbers in octal or hexadecimal.

Some sample formats:

- * `-gen-73to80-fmt="ABC%5d"`

In the first three lines the following would be inserted to columns 73 to 80:

```
ABC    1
ABC    2
ABC    3
```

- * `-gen-73to80-fmt="%06dYZ"`

In the first three lines the following would be inserted to columns 73 to 80:

```
000001YZ
000002YZ
000003YZ
```

There must be exactly one `%d` specifier in the format string. It is replaced with the generated line number. This number is equal to `col73to80.start + (generated_line_number - 1) * col73to80.step`.

If `gen_col73to80_fmt` is empty, then nothing is printed in columns 73 to 80.

Default: null string.

- `int gen_col73to80_start`

Configuration parameter: Number from which to start the numbers placed in columns 73 to 80.

Default: 1.

- `int gen_col73to80_step`

Configuration parameter: Amount by which the numbers placed in columns 73 to 80 are incremented from one line to the next.

Default: 1.

- `int debug_gen`

Configuration parameter: code generation debugging level. Changes from 0 for "no debugging" to 3 for "maximum debugging".

Default: 0.

- `int debug_posn`

Configuration parameter: position assignment debugging level.

Default: 0.

6.2 Code Generation Table

Code generation is controlled by Cobol Code Generation Table.

In this table there is exactly one entry for every Cobol Program Tree operation. List of all expression operations is given in the file `opers.h` which is generated from file `cobol.tbl`.

Each table entry of type `SctGenTblElt` has the following members:

- `SctGenTblElt()`

Default constructor.

- `SctGenTblElt(SctOper i_code, int (*i_pCond)(SctExpr *e), char *i_text)`

Constructor.

- `~SctGenTblElt()`

Destructor.

- `SctOper code`

Operation Code for which code is generated.

- `int (*pCond)(SctExpr *e)`

Pointer to a boolean Conditional Code Generation Function that for any expression node `e` tells if this code generation table entry should be applied to `e`.

When Conditional Code Generation Function is present, you can have more than one entry for a single operation.

- `char *text`

Cobol text to be generated for this operation.

This text has a number of special characters that facilitate argument substitution and text formatting.

The actions of the special characters are described below.

- `int16 primary`

Flag, used in composite table only, computed: tells whether this entry is from the primary table.

- `SctGenTblElt *next`

Link to the next element with the same code. Since we have `pCond`, one code may have more than one entry in the table. Used in composite table only, computed.

6.3 User-defined and Default tables

The code generation table is merged from the two tables:

- * User-defined Code Generation Table. This table has a precedence over the Default Table. You let the system know where your table is stored by assigning address of your table to the variable declared as `SctGenTblElt *SctPrettyPrinter::gen_table_primary`. If you do not assign this variable, then SCT assumes that there is no user-defined table, and therefore only default table will be used for code generation.
- * Default Code Generation Table. If code generation entry is not found in User-Defined table then Default Table is used.

Code Generation Algorithm. In `PrettyPrint` we use the following code generation algorithm:

Given: Expression `e`; Stream `ost`; code generation table `*gen_table`.

In table `*gen_table` find entry `g` such that:

`g.code==e.Op()` and `g.(*pCond)(e)` is True.

If found, write out `g.text` performing

formatting actions when special characters are encountered.

6.4 Formatting Actions

The following formatting actions are available:

- `$n`

Print out n -th child of this node. First child (argument) has number 1.

For instance, `$2` means print out 2nd child, or, in `SctExpr` terms, print out `e->Args(1)`.

- `^x`

Save current position as tabulation position.

- **\$x**
Move to tabulation position saved in `^x`.
Please note that `^x` should precede `$x`.
- **\$pn**
Move to preset tabulation position number `n`, where `n` is an integer number.
For instance, `$p2` moves us to preset tab position number 2 which by default is 24.
Preset tab position values are stored in `gen_tabs`.
- **{`$n?`text1: text2}**
Conditional expression.
If `n`-th child exists (that is, `e->Args(n-1)` is not NULL), then continue with `text1` and upon reaching `:` skip to `}`.
If `n`-th child does not exist, then skip to `:` and continue with `text2`.
The "Else" branch `:text2` is optional.
- **|**
Subdivision separator.
Text for every operation can be separated into *subdivisions* by `|` character.
The pretty-printing of Cobol code is done in two passes:
 - * Measure the lengths of all subdivisions for all operations in the Program Tree.
 - * If operation will not fit on the remaining part of the output line, break it into subdivisions along `|` characters.
 The text can be broken only where `|` or other separators appear.
When the break occurs, 1st subdivision remains on the current line, but subdivisions from 2nd to last are printed on the next line with indentation increased by one.
- **&**
Subdivision separator that does not advance indentation.
Same as `|` separator, but it does not advance indentation on the next line if subdivision break occurs.
- **%**
Word separator.
Printing out huge Cobol expression so that everything fits into columns 8 to 72 is a complicated optimization task. On a rare occasion, formatting algorithm cannot break all the subdivisions in a manner such that all characters fit in 8 to 72 column range.
In this case the output line overflows and it is continued on the next line. The dash (`-`) character is put to the indicator area of the next line, if it is necessary to preserve continuity of a broken word.
Often it is the case that only the last word on the line would not fit, but we still do not want to make this word a separate subdivision, because it would require printing every subdivision of this operation on a new line.

In this case we use % separator to separate this last word off.

A general recommendation on how to avoid line overflows would be: separators | and & should appear in the generated text with sufficient density, so that “official” break may occur.

- \n

Print new-line character and shut down subdivision formatting.

- \r

Shut down subdivision formatting.

- \t

Move to the next indentation position.

Indentation step is the value of parameter `gen_indent_step`.

Indentation position 0 translates to column 8 (Area A) of output text.

Indentation position 1 translates to column 12 (Area B) of output text.

Indentation position n where $n \geq 2$ translates to column $(n - 1) * \text{gen_indent_step} + 12$.

However, if boolean `gen_observe_ab_rules` is set to False, then Area B is not honored and for any $n \geq 0$ indentation position n translates into column $n * \text{gen_indent_step} + 8$ of the output line.

Also if column number becomes larger than `gen_max_indent` then indentation stops advancing. Indentation may get out of control for very deeply nested programs and therefore it needs to be stopped somewhere. Parameter `gen_max_indent` sets the limit for indentation.

- \b

Move to the previous indentation position.

This is a reversal of \t.

- \:

Print special character :, do not interpret it.

Same is true for any special character c: \c just prints out character c.

6.5 SourcePrint Overview

SourcePrint is a new capability first implemented in CobolTransformer 3.1.

It allows you to fully restore the unchanged parts of the Cobol program, while using the full power of the tree-based program transformation engine.

Our general sequence of steps needed to perform a program transformation is fairly standard: (1) parse the program making it a Program Tree, (2) transform the Program Tree, and (3) generate the converted Cobol code from the transformed Program Tree.

The only problem with this approach is that the Cobol code generated from the Program Tree usually differs from the original source code. The differences are minor – words appear in different columns, some optional words may be skipped or added.

These differences are result of a process of automatic code generation which cannot exactly reproduce the original program. While these differences may seem insignificant, they do not allow

to use `diff` or any other automatic file comparison utility to *independently verify* what lines were changed by the converter. This is because the little differences introduced by beautification make every line look different to these diff utilities.

Enter the *SourcePrint*. We store additional source token position data in the tree nodes. This data allows us to generate Cobol code that does not differ from the original code – down to a single byte – for the tree fragments that were not changed by transformations.

Code that is generated from the new tree nodes is generated in the beautified mode as was the case in the SCT versions prior to 3.1.

What does it mean for developer using CobolTransformer? It means that Cobol converters that use SourcePrint are easily and inexpensively *verifiable*. That is, upon finishing the conversion, you can run line diff utility that compares original directory with the full source of your application to the directory with the converted source. The generated difference logs are guaranteed to be minimal and reflect only changes made by the converter (and not the changes introduced by beautification or other parsing/printing side effects).

6.6 How to Use SourcePrint

SourcePrint is easy to use. In fact, addition of SourcePrint resulted in addition of very few functions to the Program Tree or PrettyPrinter APIs.

Every node in a Program Tree has certain data that connects this node to the original Cobol source. We call this data **Source Coordinates**.

There are several ways in which you can interact (implicitly or explicitly) with the SourcePrint and source coordinates while working with the Program Tree:

- * If a new node is created on the tree, this node does not know its position in source, because it never existed in the original Cobol source.
- * If a tree node that knows its source coordinates is moved to a different place in the tree, this node preserves its source coordinates. It means that you can move the whole sections, paragraphs, statements and other program units (any subtree made of nodes) and still fully preserve their formatting.
- * If a tree node is deleted, it goes away together with its source coordinates.
- * It is possible to order the existing node to “forget where it came from” by calling function `SetFromSrc(FALSE)` for this node.
- * If you call function `ResetFromSrc()` for this node then the whole subtree that starts at this node forgets its source coordinates.

This function is usually applied to subtrees that were transformed and now cannot be beautifully formatted, because some of the nodes on a transformed subtree still remember where they came from and others don't.

Source-resetting such a subtree results in beautified formatting for this subtree, while the rest of the tree retains its original look.

Please note that SourcePrint can always produce a syntactically correct Cobol program from the correct Program Tree, even if you never use `ResetFromSrc()` and feed the transformed tree directly to SourcePrint.

- * If you just need to know a position of this node (for instance, you may need it in an error message that says where in the source the error happened), you need to call function `SctTxtPosn`

`SrcPosnBeg()` on this tree node and it will return the starting position of this node in the source.

- * When Program Tree is pretty-printed or source-printed, the member `SctTxtOffsRange gen_posn` is set to the range of byte offsets (offsets are counted from the beginning of generated text) that covers this node in the generated text.

The returned range data can be used, for instance, to highlight the text that corresponds to a given tree node.

Chapter 7

Query Language

Query Language is used to construct dynamic queries. Dynamic queries are used to search Program Tree for a node that satisfies criteria specified in the query.

Together with macros queries form a foundation for on dynamically constructed transformations are built.

7.1 Query Language Reference

Introduction

When reengineering program in computer language such as Cobol, we often need to locate certain units of program that satisfy criteria that we determine.

Example 1: For instance, we may be interested in seeing all CALL and CHAIN statements for a given program, because these statements show what external programs are called from this program.

In QL this query is written as: `E.OP==STMT_CALL OR E.OP==STMT_CHAIN` or even (simplified version): `STMT_CALL OR STMT_CHAIN`

Example 2: Or we may need to find all data item declarations that belong to FILE SECTION or LINKAGE SECTION and that have VALUE clause. It would be useful if we are migrating from the Cobol compiler that allows VALUE clauses for such data items but treats them as documentary to the Cobol compiler that does not allow VALUE clauses in such data items at all.

In QL this would be:

```
E.OP=DECL_DD_ENTRY AND
(E.ISDESCOF(SECTION_WORKING_STORAGE) OR
E.ISDESCOF(SECTION_LINKAGE)) AND
E.ARG(DD_VALUE) != NULL
```

Query language described in this document is used
to formulate such search criteria.
Query is a predicate calculated on Program Tree node:

logical_value = Query (e)

So Query receives Cobol Program Tree node e as an argument and
it returns True if node e satisfies condition encoded in the Query.
The Query returns False if condition is not satisfied by e.

BNF used for QL syntax description

We use Backus Naur Form (BNF) to describe syntax
of Query Language (QL).

The Query consists of tokens.

Non-terminal tokens appear as all lower-case words in BNF rules.

Terminal tokens appear as all upper-case words in BNF rules and
they are enclosed in single quotes like this one: 'NOT'.

Words in query that match terminals in case-insensitive way,
that is, instead of 'NOT' one can use 'not', 'NoT', 'NOt' and so on.

Query Library File

Queries contain a number of characters that have
special meaning in OS shells (space , = ! ~ etc).
Therefore it makes sense to have queries loaded from files
in addition to specifying them in command line.

Since most queries are relatively short,
we can have more than one query into a file.
It also improves query management.

So, several queries may be put together in a library file.
The queries are not compiled in the library,
they appear in the source form.

In the library of queries every query must be named.

However, if file has only one query, query name may be skipped.

Every query in a query library file has header line that has format:

QUERY query-name

Here the word QUERY must be the first word on the line.

The body of the query starts on the line that immediately follows the query header line and ends on the line that immediately precedes the next query header line or end of file.

The individual query in a library is referenced by query-locator described below.

Query Lexical Rules

Query lines that starts with characters '#' or '*' are comment lines that are ignored.

If two slashes '/' are found in the line, these slashes and text after them (till the end of line) is a comment which is ignored.

White space characters ' ', '\n', '\r', '\t' are used as delimiters for tokens. That is, token is a sequence of non-whitespace characters delimited by whitespace characters.

However, when it is possible to separate tokens, whitespace between them is not required.
Any combination of name or number or string token and any of miscellaneous token is separatable.
Any other combination (name and name, name and number, number and number, etc) must be separated by whitespace.

The following tokens are used:

name: sequence of characters 'A'-'Z', 'a'-'z', '0'-'9', '-'
where first character is letter.

int-const: sequence of characters '0'-'9'.
represents integer constant.

str-const: sequence of any characters enclosed
in single quotes ' or double quotes "
with following translations inside:

```
\ " -> "  
\ ' -> '  
' ' -> ' if string starts with '  
" " -> " if string starts with "  
represents string constant
```

Miscellaneous tokens:

```
( ) = . !  
== > >= < <= ~  
+ - *  
represent various operations
```

Names of Things

Meaning of the name-like terminals:

**** query-locator**

The individual query in a library is referenced by query-locator string that has the following format:
file-name:query-name

Here file-name is a regular UNIX/MSWIN file name that can contain characters 'a'-'z', 'A'-'Z', '0'-'9', '.', '\', '/', '-'.

It also may contain character ':' in the 2nd or 3rd column of file-name if file-name starts with one of the following

[a-zA-Z]\:

[a-zA-Z]:

that is, when file-name starts with WIN32 drive designator.

It means that we disallow file-name to be one-character name with no extension.

Examples of correct query-locators:

cobc\qry.lib:qry-member

d:\cobolc\sct\QUERY-EXAMPLE.qry:my_query

/volume/dir/qrys/my-qry

**** oper**

Oper is a name-like token that represents symbolic Operation Name. Every node in Program Tree has an operation that identifies the type of node. List of all Program Tree operations is available from

the CobolTransformer Program Tree Manual.

Operation name is case-insensitive.

**** clause-index**

Clause-index is a name-like or integer-like token that represents index of child in the positional Program Tree node.

Children of every node in Program Tree are numbered from 1 to N, where N is the number of children in the node.

For positional nodes every child represents a clause, so it is convenient to have symbolic names for this clauses. Symbolic clause-index is such a name.

The list of symbolic clause names is available from the CobolTransformer Program Tree Manual.

Integer child index can be used instead of symbolic child index for positional nodes with unnamed children or for children of list nodes which are 'naturally' unnamed.

Clause name is case-insensitive.

**** node-var**

Node-var is a name-like token that represents a name of node variable. Each node variable must be explicitly declared before its use.

**** value-var**

Value-var is a name-like token that represents a name of value variable. Each value variable must be explicitly declared as either string variable or integer variable.

Therefore, there are three distinct types of variables in QL:

- Node Variable: points to Program Tree nodes,
- Integer Value Variable: holds integer value,
- String Value Variable: holds string value.

Individual Query

```

query -- query
node  -- expression that points to a Program Tree node
value -- integer or string value (constant or extracted from a node)

query: 'TRUE'
      | 'FALSE'

      // node reference points to non-existing node (node reference is NULL) or
      // pointed to node is NULL node.
      | node '==' NULL

      // node reference points to existing node (node reference is not NULL) and
      // pointed to node is not NULL node.
      | node '!=' NULL
      // equivalent to: node != NULL
      | node

      // equivalent to: node.OP == oper
      | node '==' oper

      // equivalent to: node.OP != oper
      | node '!=' oper

      // Return True if 1st and 2nd node expressions point to the same node
      | node '==' node

      // Return True if 1st and 2nd node expressions point to different nodes
      | node '!=' node

      // return True if node is a descendant of node with operation oper
      | node '.' 'ISDESCOF' '(' oper ') '

      // equivalent to: E.ISDESCOF(oper)
      | 'ISDESCOF' '(' oper ') '

      // Return True if this node originates from Cobol source.
      // Return False if this node was generated (does not come from the source).
      | node '.' FROMSRC

      // Return True if 1st operation is equal to the 2nd operation
      | oper '==' oper

      // Return True if 1st operation is not equal to the 2nd operation
      | oper '!=' oper

      // equivalent to: E.OP == oper

```

```

| oper

    // is 1st value equal to 2nd value, case-sensitive strings and integers
| value '==' value

    // is 1st value equal to 2nd value, case-insensitive strings
| value '==~' value

    // equivalent to: value ==~ value
| value '~' value

    // is 1st value not equal to 2nd value, case-sensitive strings and integers
| value '!=' value

    // is 1st value not equal to 2nd value, case-insensitive strings
| value '!=~' value

    // is 1st value >= 2nd value, case-sensitive strings and integers
| value '>=' value

    // is 1st value >= 2nd value, case-insensitive strings
| value '>=~' value

    // is 1st value > 2nd value, case-sensitive strings and integers
| value '>' value

    // is 1st value > 2nd value, case-insensitive strings
| value '>~' value

    // is 1st value <= than 2nd value, case-sensitive strings and integers
| value '<=' value

    // is 1st value <= than 2nd value, case-insensitive strings
| value '<=~' value

    // is 1st value < 2nd value, case-sensitive strings and integers
| value '<' value

    // is 1st value < 2nd value, case-insensitive strings
| value '<~' value

    // does 1st value match pattern from the 2nd value
| value 'LIKE' value

    // Return True if query is False.
    // Otherwise return False.
| 'NOT' query

```

```

| '!'    query

    // If 1st query is False, return False.
    // Otherwise evaluate 2nd query and return its value.
| query 'AND' query
| query '&&' query

    // If 1st query is True, return True.
    // Otherwise evaluate 2nd query and return its value.
| query 'OR'  query
| query '||'  query

    // return value of query
| '(' query ')';

    // call query specified by QUERY_LOCATOR with arg node
| query-locator '(' node ')';

```

Query takes in the main variable which is reference to Program Tree node called E. Based on node E, Query must compute logical True/False value and return it.

Query may refer to variables other than E. These variables:

- May be declared in the query.
Then they are called Query Variables.
- May be declared outside of query.
Then they are called Free Variables.

Main variable, Query Variables and Free Variables all point to Program Tree nodes. References to them appear as non-terminal 'node' in Query BNF.

When Query is evaluated, it receives values of all variables, including main variable and free variables.

Query variables are assigned value inside the query.

Comparison operators and assignment operators:

We follow C, C++ and Java tradition and use

'=' as assignment operator and
'==' as equal-to comparison operator.

LIKE operator matches its 1st argument against the pattern taken from the 2nd arg. The RX package is used for pattern matching, and format of pattern string is described there.

Empty string query is equivalent to 'TRUE' query, that is,
to query that returns True for any Program Node presented to it.

```

node: // Return value of the main node variable
      'E'

      // Return the Root variable.
      // This is a root of the Program Tree in which search occurs.
| 'ROOT'

      // Return value of the previously declared node variable
| node-var

      // assign node's value to the previously declared variable var-node
| node-var '=' node

      // declare node variable var-node and assign node's value to it
| 'NODE' node-var '=' node

      // return parent of the node
| node '.' 'PARENT'

      // equivalent to: E.PARENT
| 'PARENT'

      // return child/argument number clause-index of the node
| node '.' 'ARG' '(' clause-index ')'
| node '.' 'CHILD' '(' clause-index ')'

      // equivalent to: node.ARG(CLAUSE_INDEX_CONSTANT)
| node '.' CLAUSE_INDEX_CONSTANT

      // equivalent to: E.ARG(clause-index)
| 'ARG' '(' clause-index ')'
| 'CHILD' '(' clause-index ')'

      // Find 1st occurrence of operation oper among arguments of node.
      // If not found, return NULL.
| node '.' 'FIND' '(' oper ')'

      // equivalent to: E.FIND(oper)
| 'FIND' '(' oper ')'

      // Return Linked-to node for node thatfor which HasLink() is True.
      // Returns UNDEFINED for all other nodes.

```

```

    // Returns NULL if object has no definition.
| node '.' 'LINK'

    // Enclosing statement for the node
| node '.' ENCL-STMT

    // Enclosing section for the node
| node '.' ENCL-SECTION
;

```

Non-terminal node represents reference (pointer) to Program Tree node. These pointers may have NULL values for different reasons. For instance, when data item declaration does not have a certain clause, the child that is responsible for this clause is NULL.

If attempt is made to dereference NULL node pointer (for instance, in node.PARENT operation), the dereferencing does not occur. If node pointer is expected then NULL pointer is returned.

E represents main variable of a query.

Query Variables that are declared in the query are visible only in this query. If variable needs to be visible outside of the query (that is, in Scripting Language), it must be declared in SL.

```

value: // Integer constant
| int-const

    // String constant
| str-const

    // Image of the value stored in the node (string)
| node '.' 'IMAGE'

    // Normalized Image of the value stored in the node (string)
| node '.' 'VALUE'

    // Integer value of the node (valid only for INT_NUM_CONST node)
| node '.' 'INTVALUE'

    // Number of arguments/children of the node
| node '.' 'ARGNO'

    // Return value of integer/string variable var-value

```

```

| var-value

    // Assign value to previously declared integer/string variable var-value
| var-value '=' value

    // Declare integer variable var-value and assign value to it
| INT var-value '=' value

    // Declare string variable var-value and assign value to it
| STR var-value '=' value

    // Return length of string value
| value '.' 'LENGTH'

    // Return substring of string value that starts
    // at integer position value-1 (first character has position 0) and
    // has length of integer value-2
| value '.' 'SUBSTR' '(' value-1 ',' value-2 ')'

    // Return position of 1st occurrence of string value-1 in value.
    // First position in the string has number 0.
| value '.' 'pos' '(' value-1 ')'

    // Return position of i-th occurrence of string value-1 in value.
    // First position in the string has number 0.
| value '.' 'pos' '(' value-1 ',' i ')'

    // add two integers or concatenate two strings
| value '+' value

    // subtract two integers
| value '-' value

    // multiply two integers
| value '*' value

    // divide two integers
| value DIV value

    // compute remainder of division for two integers
| value MOD value

    // lower-case the string
| value '.' LOWER

    // upper-case the string
| value '.' UPPER

```

;

If attempt is made to dereference NULL node pointer
(for instance, in node.OP operation), the dereferencing does not occur.
If operation or value are expected to be returned as a result
of dereferencing (as in node.OP), special UNDEFINED value is returned.

The UNDEFINED value behaves by the following rules:

```
value == UNDEFINED -> FALSE
value != UNDEFINED -> TRUE
value > UNDEFINED -> UNDEFINED
value >= UNDEFINED -> UNDEFINED
value < UNDEFINED -> UNDEFINED
value <= UNDEFINED -> UNDEFINED
value + UNDEFINED -> UNDEFINED
value - UNDEFINED -> UNDEFINED
value * UNDEFINED -> UNDEFINED
value DIV UNDEFINED -> UNDEFINED
value MOD UNDEFINED -> UNDEFINED
value && UNDEFINED -> UNDEFINED
value || UNDEFINED -> value
! UNDEFINED -> UNDEFINED
```

Operation priorities from highest to lowest:

```
. 'FUNC'
* / DIV MOD
+ -
comparison opers
=
```

operations are computed left to right, that is
 $a * b * c = (a * b) * c$

Developer API

If you purchased a complete CobolTransformer distribution,
see file sct/query.h for detailed instructions
on how to use Developer API for Query Language.

Not implemented

As of Version 3.2.1 of QL, the following constructs are not implemented:

- * Query: QUERY_LOCATOR '(' node ')'
- * Query Variables
- * Free Variables

7.2 Query Language Examples

```
#
# This is Query Library file that contains examples of useful queries.
# These queries are used in conversion tools based on CobolTransformer.
#

#
# Find TRANSFORM statements
#
Query TransformStatement
  stmt_transform

#
# Find declaration of data item that has VALUE clause
#
Query DDhasValueClause
  DECL_DD_ENTRY and
  arg(DD_VALUE) != NULL

#
# Find data item declaration with no PICTURE clause
#
Query DDnoPICclause1
  e.op==DECL_DD_ENTRY and
  e.arg(DD_PICTURE) == NULL

Query DDnoPICclause2
  DECL_DD_ENTRY and
  Child(DD_PICTURE) == NULL

#
# Find PLUS/MINUS operations used in table references in the following way:
#   TABLE (index - +1) or TABLE (index - -1) or
#   TABLE (index + +1) or TABLE (index + -1)
#
```



```

Query TableRefIndexPlusMinusOrMinusPlus
  (OP_ADD_IX or OP_SUB_IX) and
  parent==SUBSCRIPT_LIST and
  parent.parent==TABLE_ELT and
  e.ArgNo==2

#
# Find File Declaration (FD) with REPORT clause and
# no clause that specifies record size (RECORD CONTAINS or RECORD IS VARYING)
#
Query FDwithReportClauseAndWithNoRecordContains
  DECL_FD and
  e.Arg(FDH_ENTRY).Arg(FD_REPORT) and
  e.FDH_ENTRY.FD_RECORD_SIZE == null

#
# Find data item declarations that have VALUE clause and
# belong to WORKING-STORAGE or LINKAGE sections.
#
Query DDhasValueInWorkingOrLinkageSection
  E.OP==DECL_DD_ENTRY AND
  (E.ISDESCOF(SECTION_WORKING_STORAGE) OR
  E.ISDESCOF(SECTION_LINKAGE)) AND
  E.ARG(DD_VALUE) != NULL

#
# Find all code which affects data item "WSS",
# either directly or thru REDEFINES.
# See the power of QL!
#
Query WhatAffectsDataItem
( ( e==NAMED_OBJ_DEF && e.value~"WSS" && // Definition
  // Save section flag: only one of assignment
  // will be executed.
  ((e.isdescf(section_working_storage) && int sect=1) ||
  (e.isdescf(section_linkage) && int sect=2)
  )
  &&
  // save beginning and end of definition
  int base=e.parent.dd_offset_global.intvalue &&
  int len=base+e.parent.arg(dd_length).intvalue
  )
  ||
  ( e==NAMED_OBJ_USE && // Usage
  // In the same section?
  ((e.def.isdescf(section_working_storage) && sect==1) ||
  (e.def.isdescf(section_linkage) && sect==2)
  )
  )

```

```

    ) &&
    // Overloaded?
    ( int head=e.link.parent.dd_offset_global.intvalue and
      int tail=head+e.link.parent.dd_length.intvalue &&
      ( (head>=base && head<=len) || (tail>=base && tail<=len) )
    ) &&
    // Does this USE belong to receiving part of
    // MOVE, ADD, SUB, MUL, DIV?
    ( e.isdescof(ex_list) || e.isdescof(ex_ident_list) ||
      e.isdescof(ex_arith_div_remainder)
    )
  )
)
)

```

7.3 Query API: SctQuery

You need to compile query text before using query for searching the tree. Compilation is very fast and it is performed by `SctQuery` object.

`SctQuery` object has the following public members:

- `SctQuery()`
Default constructor.
- `SctQuery(const SctQuery &q)`
Copy constructor.
- `~SctQuery()`
Destructor.
- `void Clear()`
Clear this query, make it empty one.
- `SctErrCode Parse(const VString &s, SctVarList *varlist, VString &err_msg, int &err_pos)`
Parse query text from string `s` into query internal representation stored in this object. `*varlist` is a list of external variables (used in SL). If `varlist` is NULL, only variable `E` is allowed.
If query was successfully parsed, `SCT_OK` is returned. If query contains an error, error message is written to `err_msg`, error position in `s` is written to `err_pos` and error code is returned.
- `SctErrCode Parse(istream &ist, SctVarList *varlist, VString &err_msg, int &err_pos)`
Same as above, but parse istream `ist` instead of string.
- `int IsTrue(SctExpr *e) const`
Return True, if node `e` satisfies this query. Otherwise return False.
- `void PutTic(ostream &ost)`
Output this query to ostream `ost` in TIC form.

- `SctErrCode GetTic(istream &ist, VString &err_msg)`
Get this query from istream `ist`.
- `friend inline ostream &operator <<(ostream &ost, const SctQuery &q)`
- `void Print(ostream &cout) const`
Print this query in human-readable form to ostream `ost`.
- `VString query_txt`
Query text.

Chapter 8

Transformation

Transformation is an object that searches and/or transforms Program Tree. Transformations follow certain conventions that allow CobolTransformer system relieve a developer from many chores related to Program Tree transformation.

Conveniences that transformations bring to us:

- Easy Reuse

If you use transformations to program your Cobol conversions, you would get an easy access to the library of existing transformations. Also following our conventions would allow you to put your transformations into the similar library, so that when you start work on your next conversion project, transformations from the previous project can be reused.

- Automatic Configuration

Transformations may have configuration parameters. Each configuration parameter may have an option descriptor attached to it. This descriptor tells the system how a given parameter can be changed by the end user both from command line and from tabbed dialogues belonging to visual tool.

- Debugging and Diagnostics

Transformation engine implements all necessary debugging and diagnostics for your transformations, so that you don't have to.

- Save and Load transformations

You can save and load any instantiated transformation.

8.1 Transformation ID

Transformation ID is used to reference transformation that exist both externally in files and internally in memory. One transformation may have several SctTransIDs pointing to it.

Transformations can be owned by the Transformation ID or not. Ownership implies clean-up duties: when TransID is deleted, transformation owned by it is also deleted.

Transformations can be associated with transformation storage that manages saving and loading of transformations from transformation file path.

SctTransID class has the following members:

- `SctTransID()`
Default constructor.
- `const VString &i_name`
Construct Transformation ID that points to transformation named `i_name`.
- `SctTransform *i_ptr, int i_own = FALSE`
Construct Transformation ID that points to in-memory transformation that has address `i_ptr` and ownership property `i_own`.
- `~SctTransID()`
Destructor.
- `void DellfOwn()`
Delete a referenced transformation if it is owned.
- `void PutTic(ostream &ost) const`
Put this in TIC form.
- `SctErrCode GetTic(istream &ist, VString &err_msg)`
Get this in TIC form.
- `friend ostream &operator << (ostream &ost, const SctTransID &tid)`
- `virtual void Print(ostream &ost) const`
Print this transformation to stream `ost`.
- `SctErrCode Load(SctTokenList &msgs)`
Load transformation identified here to memory from default transformation storage. If load errors occurred, error message to `msgs` and return error code. If transformation identified here is already loaded into memory, just set `ptr` to point to this transformation.
- `SctTransform *Ptr()`
Return pointer to the transformation identified by this `TransID`. If transformation is owned, just return memorized pointer – it cannot change. If transformation is not owned, ask the storage to compute the pointer, possibly loading the referred transformation.
- `SctTransform *ReadPtr() const`
Return pointer to the transformation identified by this `TransID`. If transformation is owned, just return memorized pointer. If transformation is not owned, return `NULL`.
- `VString name`
Transformation name. If this transformation belongs to storage, this name uniquely identifies transformation in the storage.
- `int own`
If `own` is `True` the transformation pointed to in this `TransID` is owned by this `TransID`.
- `static SctTranStorage *stor`
Default storage used by `Ptr()` for looking up transformations.

8.2 Transformation storage

Each transformation in a storage is identified by the name. One transformation occupies one file and transformation name is equal to file name plus `".sct"` extension. Transformation name is unique on storage file path.

Transformation storage consists of 2 parts: memory space and file space:

- * Some transformations exist only in file space, they were not called for and therefore not loaded into memory.
- * Other transformations exist both in file space and memory space, moreover memory version can be different from file version, when, for instance, the memory was changed and not yet saved.
- * Some transformations exist only in memory space and not in file space. This should be rare situation which applies mostly to newly created transformations not yet saved to file.

When a transformation exists both in memory space and in file space, all operations apply to memory-based transformation which is considered to be current version, while file transformation is a previous version.

After you create a transformation in memory, you need to explicitly register it with storage if you want this transformation to be part of the storage. Before deleting transformation from memory, you need to explicitly delete it from the storage to which it belonged. Files being added or deleted on the storage path automatically affect transformation thru the file system but we still recommend to use `transtorage` functions to ensure correctness.

`SctTranStorage` class has the following members:

- `SctTranStorage(const VStringList &i_trans_dirs = VStringList())`
Construct transformation storage with file path `i_trans_dirs`.
- `~SctTranStorage()`
Destructor. Since loaded transformations do not belong to the storage, do not delete anything here.
- `int FindInMem(const VString &name, SctTransform **p_trans = NULL)`
Find transformation named `name` in memory space of this storage. If transformation is found in memory space, pointer to it is written to `*p_trans`. Return `True` if transformation was found. Return `False`, if not found.
- `int FindInFile(const VString &name, VString *p_dir_name = NULL)`
Find transformation named `name` in file space of this storage. If transformation is found in file space, its directory name is written to `*p_dir_name`. Return `True` if transformation was found. Return `False`, if not found.
- `int Find(const VString &name, SctTransform **p_trans = NULL, VString *p_dir_name = NULL)`
Find transformation named `name` in this storage, in memory space first. If transformation is found in memory space, pointer to it is written to `*p_trans`. If transformation is found in file space, its directory name is written to `*p_dir_name`. Return `True` if transformation was found. Return `False`, if not found.
- `SctErrCode AddToMem(SctTransform *trans, VString &err_msg)`

Add transformation ***trans** to this storage as memory-only transformation. Check that the new transformation is not already present in this storage. If error occurs, write it to **err_msg** and return error code.

- **SctErrCode CopyFromMemToFile(SctTransform *trans, VString &err_msg)**

Copy transformation ***trans** to this storage file space by saving it. If error occurs while saving, write error message to **err_msg** and return error code.

- **SctErrCode CopyFromFileToMem(const VString &name, SctTransform *&trans, VString &err_msg)**

Copy transformation named **name** from file space to memory space. Write pointer to loaded transformation to **trans**. If transformation is already loaded into memory, just return **SCT_OK**. If load errors occurs, write error message to **err_msg** and return error code.

Ownership of the loaded transformation is transferred to the caller. So whoever loaded this transformation, must delete it at some point.

- **SctErrCode DeleteFromMem(const VString &name, int verify, VString &err_msg)**

Delete transformation named **name** from memory space. If **verify** is **True**, make sure that we indeed deleted something. If error occurs, write error message to **err_msg** and return error code.

- **SctErrCode DeleteFromFile(const VString &name, int verify, VString &err_msg)**

Delete transformation named **name** from file space, If **verify** is **True**, make sure that we indeed deleted something. If error occurs, write error message to **err_msg** and return error code.

- **SctErrCode Delete(const VString &name, int verify, VString &err_msg)**

Delete transformation named **name** both from memory space and from file space. If **verify** is **True**, make sure that we indeed deleted something. If error occurs, write error message to **err_msg** and return error code.

- **SctErrCode Rename(const VString &old_name, const VString &new_name, const VString &new_dir_name, int delete_new, int update_refs, VString &err_msg);** Rename/move transformation named **old_name** to transformation named **new_name** in directory **new_dir_name** in this storage. If **old_name** and **new_name** are the same, just save the transformation. Otherwise if transformation **new_name** exists: if **delete_new** is **True** then delete **new_name** otherwise report error. If transformation **old_name** does not exist, report error. If **update_refs** is **True**, update all TransIDs pointing to the renamed transformation.

- **static VString FullFileName(const VString &dir_name, const VString &file_name)**

Produce full transformation file name from directory name and transformation name.

- **VStringList trans_dirs**

Transformation Directory path. This is list of directories in which transformations that can be browsed or loaded are located.

- **int loaded_no;**

SctTransform *loaded[SCT_LOADED_TRANS_NO]; Array of transformations that are currently loaded into memory by this storage.

8.3 Transformation

In general transformation is a black box that transforms/searches the Program Tree. Search and Transform functions of the transformation are invoked through the official interface functions `CanStartAt()`, `AppllyOnce()`, `ApplyAll()`. Transformation algorithm itself is contained in `SearchTransform()` function, and `CanStartAt()`, `AppllyOnce()`, `ApplyAll()` functions are mere repackaging of `SearchTransform()`.

`SctTransform` is an abstract class. All concrete transformations are inherited from this class. Concrete transformations are allowed to have their own data members. These data members can be used for configuration of transformation or to store internal data that should be retained between invocations of `SearchTransform()`.

Static Data. Transformation class behaves as a black box, but it declares what it does in the class description `ClassDescr()`. The static data of the transformation such as `ClassDescr()` or `SearchTransform()` function cannot be changed, copied, loaded, saved by the user.

Each transformation class is uniquely identified by its name.

Dynamic Data. However, transformation is not entirely a black box creature. Some aspects of it are in the open and changeable by user:

- * Transformation name and its instance description.
- * Configuration parameters, if present.
- * References of the transformation. Transformation may refer to a number of other transformations. The referred-to transformations are usually called by this transformation (the exact semantics is determined by transformation class). Each referred-to transformation may be enabled or disabled. This transformation usually takes an obligation not to call disabled transformations.
- * More on dynamic data below.

Dynamic transformation data can be changed, copied, loaded, saved by the user.

`SctTransform` class has the following members:

8.3.1 Constructors, destructors, creators, memory management

- `SctTransform()`
Default Constructor.
- `SctTransform(const VString &i_name, const VString &i_inst_descr, int (*i_pCutOff)(SctExpr *e, SctExpr *e0) = NULL, SctErrCode (*i_pTransWalker)(SctExpr *e, void *arg_block) = SctTransWalker)`
Constructor.
- `static void *operator new(size_t s)`
Custom `new` that records allocated transformations.
- `virtual SctTransform *Copy() const = 0`
Return a copy of this transformation. Since C++ does not have virtual constructors, do not use copy constructor to copy a transformation.

- virtual `~SctTransform()`
Delete all referenced transformations that are not in storage.
- static void operator delete (void *p, size_t size)
Custom delete that de-records deleted transformation
- static void DeleteAll()
Delete all allocated transformations.

8.3.2 Transformation class properties and description

- virtual `SctRefsProps RefsProps() const`
Return properties of the references of this transformation. Transformation tells us what can be done with references. By default we cannot have references.
- virtual `int ChangesTree() const`
Tell whether this transformation modifies the tree. If it does not, transformation can be used as a finder.
- virtual `int KillsAncestors() const`
Tell whether this transformation deletes (kills) ancestors (including immediate parents) of its current node. We conservatively assume that transformations do it. If we find that it slows process too much, then we would assume that most of them don't do it (historically they did not do it).
- virtual `VString ClassName() const`
Return name of this transformation class.
- virtual `VString ClassDescr() const`
Return long description of this transformation class.
- `VString ShortClassDescr() const`
Return short description of this transformation class. Generated from the long description returned by `ClassDescr()` cutting out text that goes after the first newline character.
- static `SctTransformDescr *descrs`
Transformation class descriptors.
On transformation registration: properly declared transformation do self-register. The registration occurs at the static global objects initialization time. Since C++ compiler does not guarantee that transformation registration caused by static objects in transformations will run after the initialization of `descrs` array (in fact, MS VC++ does the opposite), we have to initialize the `descrs` array pointer to NULL. Most compilers say that initializations to null are run before all other initializations, so starting with `descrs=NULL` almost guarantees that nullification happens before transformation registration, not afterwards.
- static `SctErrCode Register(SctTransform *tr, SctTransform *(*pcreator)(), VString &err_msg)`
Register transformation `tr` that has creator function `*pcreator`. If transformation is already registered, write error message to `err_msg` and return `SCT_ERR_TRANS_ALREADY_PRESENT`. If registered, return `SCT_OK`.

- `static SctErrCode UnRegister(SctTransform *tr, VString &err_msg)`
UnRegister transformation `tr`. Transformation is looked up by its class name. If transformation is not found, write error message to `err_msg` and return error code. If unregistered, return `SCT_OK`.
- `static SctErrCode UnRegister(int i_from, int i_to, VString &err_msg)`
UnRegister transformations with indices in `descrs` table from `i_from` to `i_to`.
- `static SctErrCode NewByName(const VString &class_name, SctTransform *&tr, VString &err_msg)`
Generate instance of a transformation given its class name.

8.3.3 Init, Config, Find and Apply operations

- `virtual void SetOptions()`
Set options template. This function must be called before you use options.
- `virtual SctErrCode SetConfig(SctParser *parser, SctPrettyPrinter *pprinter, SctTokenList &msgs)`
Make transformation configuration effective. Some transformations may have internal configuration parameters that are simply public data members. You can set them as such, but the new configuration is not guaranteed to become effective until you call `SetConfig()`.
When setting configuration you may supply transformation with information on your parser and pretty-printer settings. For instance, transformation may want to turn on certain options if certain optional dialect is specified to parser. If you do not want parser/pretty-printer to be involved, specify NULLs for them.
If configuration is not successful, error code and error message are returned.
Example: large block transformation refers to many small transformations and the large transformation presents a single interface to all subordinated transformations. When you call `SetConfig()` for the large transformation, config data is propagated down to subordinated transformations.
- `virtual SctErrCode Init(SctTokenList &msgs)`
Initialize transformation internal non-configuration data. Some transformations may store data that persists between transformation invocations by `ApplyOnce()`. `Init()` is called to reset such data to its initial state. It is recommended that you call `Init()` in constructor for the transformation to ensure that the newly constructed transformation is properly initialized. If initialization is not successful, error code and error message are returned.
Example: transformation creates data items in data division and memorizes their addresses. Each subsequent invocation adds uses of the memorized data items to the program. When we go to the next program, we should call `Init()` so that new program does not refer to data declared in the old program.
- `SctErrCode FindNext(SctTree &t, SctTokenList &msgs)`
Starting at the node that follows the current node of the tree `t` and moving forward, find node `e` in the tree `t` such that this transformation can be applied to the node `e`. If we reached end of this tree and did not find the node, set the current node to the last node of the tree and return error code.

- **SctErrCode FindPrev(SctTree &t, SctTokenList &msgs)**
Starting at the node that precedes the current node of the tree **t** and moving backwards, find node **e** in the tree **t** such that this transformation can be applied to the node **e**. If we reached beginning of this tree and did not find the node, set the current node to the first node of the tree and return error code.
- **SctErrCode FindAll(SctTree &t, SctLinkList &found_nodes, SctTokenList &msgs)**
Find all nodes **e** in the tree **t** this transformation can be applied to the node **e**. Assemble found nodes in link list **found_nodes**.
- **SctErrCode CanStartAt(SctExpr *e, int &can_start, SctTokenList &msgs)**
Determine whether this transformation can start at the node **e**. If yes, write **True** to **can_start**, otherwise write **False** to **can_start**. If errors occur while finding the answer, add the errors to **msgs** and return error code.
- **SctErrCode ApplyOnce(SctExpr *e, int transform, SctTokenList &msgs, SctExpr **pe_after = NULL)**
Apply this transformation to the node **e**. If **transform** is **False**, only search for nodes that satisfy the transformation, but do not change the tree. If **pe_after** is not **NULL**, **ApplyOnce** assigns to ***pe_after** the new current node as computed by the transformation, however this assignment is not guaranteed to happen. That is, if both caller of **ApplyOnce** and transformation called by **ApplyOnce** agree to pass on the new current node, they can do it using ***pe_after**.
If error occurs while applying transformation, error message is added to **msgs** and error code is returned. Warning and info messages may be added to **msgs** even in the case of normal completion of this transformation.
- **SctErrCode ApplyOnce(SctTree &t, int transform, SctTokenList &msgs, SctExpr **pe_after = NULL)**
Apply this transformation to the current node of the tree **t**. If error occurs while applying transformation, error message is added to **msgs** and error code is returned.
- **virtual SctErrCode SearchTransform(SctExpr *e, int transform, SctTokenList &msgs, SctExpr **pe_after = NULL) = 0**
This is a function that defines transformation behavior.
This function, when called on node **e**, checks that node **e** may serve as a starting node of this transformation.
If node **e** is not a starting point, return **SCT_RC_CVT_NOT_APPLICABLE**.
If node **e** is a starting point and flag **transform** is **False**, or transformation is finder only and does not change the tree, then transformation returns **SCT_OK** when it finds the node.
If node **e** is a starting point and flag **transform** is **True**, then transformation is performed and status code of the transformation is returned. If everything went fine, **SCT_OK** code is returned. If error occurred while searching (does not happen) or transforming (can happen), error messages are added to **msgs** and error code is returned.
- **void WillTransform(int silent)**

This function is called by transformation when it makes a decision to change the tree and reports this decision to the system. If `silent` is `False` and `SctTransform::conv_warn` is `True`, then message describing this transformation application will be added to `*curr_pmsgs`. Arguments of the current transformation are memorized in static members of `SctTransform`, so there is no need to pass them to `WillTransform`.

- `virtual SctErrCode ApplyAll(SctExpr *e, int transform, SctTokenList &msgs, SctExpr **pe_after = NULL)`

Apply this transformation to all nodes that are below node `e`. However, transformation can possibly modify the nodes that are above the current node. If error occurs while applying transformation, error message are added to `msgs` and error code is returned.

- `SctErrCode ApplyAll(SctTree &t, int transform, SctTokenList &msgs, SctExpr **pe_after = NULL)`

Apply this transformation to all nodes that are below the current node of the tree `t`. However, transformation can possibly modify the nodes that are above the current node. If error occurs while applying transformation, error messages are added to `msgs` and error code is returned.

- `SctErrCode ApplyRepeat(SctExpr *e, SctTokenList &msgs, SctExpr **pe_after = NULL)`

Iteratively apply this transformation to the node `e`. If `pe_after` is not `NULL`, assign to `*pe_after` the new current node as computed by the transformation. Keep applying transformation to the node `e` or to the node returned in `*pe_after` until transformation returns `SCT_OK`.

If error occurs while applying transformation, error message is added to `msgs` and error code is returned. Warning and info messages may be added to `msgs` even in the case of normal completion of this transformation.

8.3.4 References

- `SctTransform *FindRef(const VString &name)`

Find transformation named `name` among the referenced transformations. If not found return `NULL`.

- `int InsRef(const SctTransRef &tr, int ix)`

Insert new referenced transformation element `tr` into position `ix` of this array of referenced transformations. If `ix` is `-1` then add the new transformation element as the last one. Return index of the new reference.

- `int AddRef(const SctTransRef &tr)`

Add new referenced transformation element `tr` to the array of referenced transformations. Return index of the new reference.

- `int AddRef(SctTransform *t, int is_owned = TRUE)`

Create enabled reference that points to transformation `t` and add this new reference to the referenced transformations. This transformation is attached in owned/non-owned mode depending on flag `is_owned`. Return index of the new reference.

- **SctTransRef DelRef(int ix)**
Delete referenced transformation element from the array of referenced transformations at position `ix`. If `ix` is `-1` then detach last element of the array. Function return deleted referenced transformation element.
- **SctErrCode LoadAll(SctTokenList &msgs)**
Load all referenced transformations. If errors occur while loading transformations, error messages are added to `msgs` and error code is returned.

8.3.5 Save/Load

- **SctErrCode SaveToFile(VString &err_msg)**
Save this transformation to file specified in it by `name` and `dir_name`. If error occurs, write it to `err_msg` and return error code.
- **static SctErrCode LoadFromFile(const VString &file_name, SctTransform *&ret, VString &err_msg)**
Load transformation from file specified by `file_name` and write it to `*ret`. If error occurs, write it to `err_msg` and return error code.
- **void PutTic(ostream &ost) const**
Output this transformation in TIC form. If no such function exists at the derived class, this default function is called.
- **static SctErrCode GetTic(istream &ist, SctTransform *&ret, VString &err_msg)**
Get transformation from istream `ist` and write it to `*ret`. If parsing error occurs, write it to `err_msg` and return error code.
- **virtual SctErrCode GetTicOper(istream &ist, VString &err_msg)**
Get this transformation from istream `ist`. Used by implementation of `GetTic()`.
- **friend ostream &operator << (ostream &ost, const SctTransform &tr)**
- **virtual void Print(ostream &ost) const**
Print this transformation to stream `ost`.

8.3.6 Diagnostics and Debugging

- **static void (*pError)(const VString &s)**
Pointer to user-defined Bad Error processing function.
- **static int conv_warn**
Control flag: if `True`, warn about transformation problems such as inability to transform.
- **void Warning(const VString &msg)**
Transformation calls this function to issue a warning about transformation problem.

- `static int conv_info`
Control flag: if True, for every transformation application add to `msgs` a message that describes the applied transformation.
- `void Info(const VString &msg)`
Transformation calls this function to issue informational message.
- `static int find_msg`
Control flag: if True, add a **found the location** message to `msgs` when transformation is applied. Otherwise add regular message about transformation applied. Default: False.
- `static int debug`
Control flag: debugging level.
- `static int user_debug`
Control flag: user transformations debugging level.
- `static SctExpr *curr_e`
- `static int curr_transform`
- `static SctTokenList *curr_pmsgs`
Arguments of the currently executed `SearhTransform()` function.
- `static int curr_trans_stack_depth`
- `static SctTransform *curr_trans_stack[SCT_TRANS_STACK_DEPTH]`
Stack of transformations that are now active.
Transformation `*curr_trans_stack[curr_trans_stack_depth-1]` is the currently executed transformation.
- `static SctExprPath curr_walker_stack`
Transformation walker call stack. Used by walker to recover from transformations that delete their parents.

8.3.7 Data members

- `VString name`
Name of this transformation. Should be different from names of all other transformations in the storage.
- `VString inst_descr`
Description of this transformation instance. By default set to `ShortClassDescr()`.
- `VString ShortInstDescr() const`
Return short description of this transformation instance. Generated from the long description `inst_descr`.
- `VString dir_name`
Directory in which this transformation is stored. Set by user before saving this transformation to storage. Set by the system when transformation is loaded from storage.

- **SctOptionDescList** options
CommandLine/Visual options list. These options are used to set configuration parameters of this transformation.
- **int (*pCutOff)(SctExpr *e, SctExpr *e0)**
Pointer to walker cut-off condition. ApplyAll() walker does not walk into nodes for which this function returns **True**. The function receives two arguments: currently walked node and node at which the walk has started. Default cut-off function is no cut-off at all.
- **SctErrCode (*pTransWalker)(SctExpr *e, void *arg_block)**
Pointer to walker function. Default walker is supplied but you can replace it. Remember to call cut-off function in your custom walker.
- **int ref_elts_no**
- **SctTransRef ref_elts[SCT_TRANS_REF_NO]**
Array of transformations referenced in this transformation. This transformation usually promises to honor **enabled** flag specified in the **SctTransRefs**.
- **int is_owned**
If this flag is **True**, then some other transformation owns this transformation. It means that this transformation is read-only, because operations applied to owner may change this transformation.

8.4 Macros used in transformations

8.4.1 Declaration Macros

The macros below are used for derived transformation class declaration. They save you effort in writing up standard class regalia. Also they class with transformation auto-loader.

- **SCT_TRANS_DECL(name, cd)**
Declare transformation with name **name** and description **cd**. Transformation has no data members.
- **SCT_TRANS_DECL_BEGIN(name, cd)**
Begin declaration of transformation with name **name** and description **cd**. Used for transformations with config/internal data members.
- **SCT_TRANS_DECL_BEGIN_BLOCK**
Declare block transformation with name **name** and description **cd**. Derived from **ScvtBlock**. Blocks declared with this macro have fixed number of references which cannot be disabled.
- **SCT_DOES_NOT_CHANGE_TREE**
Declare that transformation does not change the tree. Can be used in **SCT_TRANS_DECL_BEGIN .. SCT_TRANS_END_DECL** block.
- **SCT_TRANS_END_DECL()**
End declaration of transformation.

8.4.2 Transformation Body Definition

These macros are used to define transformation body, that is, transformation's `SearchTransform()` function.

- `SCT_TRANS_FILE_REGISTER(file)`
Register transformations in C++ translation unit `file`. This macro should appear at the beginning of the C++ file that contains transformations to be registered.
- `SCT_TRANS_FULL_NAME(trans)`
Full name of the `SearchTransform()` function for transformation `trans`.
- `SCT_TRANS_BODY(trans)`
Start `SearchTransform()` function body for transformation `trans`. Also generate C++ helper class that registers transformation `trans`. Every transformation that wants to be registered must have body declared with `SCT_TRANS_BODY`.

8.4.3 Macros used in Transformation Body

These macros are used inside transformation body, in `SearchTransform()` function.

- `SCT_SEARCH_TRANSFORM_BASE(base)`
Call `SearchTransform()` of the base class transformation `base`.
- `SCT_MAKE_SURE(cond)`
If condition `cond` computed on the current node is not `True`, bail out of this transformation with `SCT_RC_CVT_NOT_APPLICABLE` code.
- `SCT_MAKE_SURE(cond, warn)`
If condition `cond` computed on the current node is not `True`, issue warning `warn` and bail out of this transformation with `SCT_RC_CVT_NOT_APPLICABLE` code.
- `SCT_MAKE_SURE_OPER(op)`
- `SCT_APPLICABLE_TO(op)`
If operation at the current node is not `op`, bail out of this transformation with `SCT_RC_CVT_NOT_APPLICABLE` code.
- `SCT_WILL_TRANSFORM()`
This transformation declares that it found eligible node and asks for permission to change the tree.
If permission is not granted (`transform` is `False`) then transformation was asked only to find a tree node and not to change the tree, therefore transformation exits with return code `SCT_OK`.
If permission is granted (`transform` is `True`) then the message that describes this decision to transform is added to `msgs`.
Transformation must not change the tree before it issues `WillTransform`.

- `SCT_WILL_TRANSFORM_SILENT()`
Same as above, but do not add message about this transformation to `msgs`. This call is appropriate for transformations that do not want user to know that they are being performed.
- `SCT_WILL_TRANSFORM_CHANGE()`
Same as in `SCT_WILL_TRANSFORM` and also set variable `change` to `True`.
- `SCT_WILL_TRANSFORM_OBJ(t,transform)`
Same as in `SCT_WILL_TRANSFORM` but do it on transformation `t` instead of on `this` transformation.
- `SCT_WILL_TRANSFORM_FOUND()`
Transformation declares that it found what it was looking for. This variety of `WillTransform` should be used only in finder transformations that do not change the tree.
- `SCT_TRANS_BODY_END`
End of the body of transformation.

Chapter 9

Transformation Library

9.1 Cobol Beautifier Transformation

- **Block ScvtCobolBeautifier**

Beautify and Cleanup Cobol code.

Configuration parameters:

- * **int initial_values**
If True, add VALUE clauses to WS data items.
- * **int norm_dd_levels**
If True, normalize data item levels, so that data items at one logical hierarchy level have the same level numbers.
- * **int add_end_stmts**
If True, Add END-IF, END-SEARCH, END-EVALUATE, END-PERFORM closing statements.

9.2 Renumbering Transformations

- **ScvtRenumberSections**

Renumber sections. New section name is generated using printf-like **section_name_fmt** and section counter that runs from **section_name_start** with step **section_name_step**.

Configuration parameters:

- * **VString section_name_fmt**
New section name format. Should have exactly one %d for section counter.
- * **int section_name_start**
Section counter start value.
- * **int section_name_step**
Section counter step.

- **ScvtRenumberParagraphs**

Renumber paragraphs. New paragraph name is generated using printf-like **para_name_fmt** and paragraph counter that runs from **para_name_start** with step **para_name_step**.

Configuration parameters:

- * `VString para_name_fmt`
New paragraph name format. Should have exactly one `%d` for paragraph counter.
- * `int para_name_start`
Paragraph counter start value.
- * `int para_name_step`
Paragraph counter step.

- **ScvtRenumberDataItems**

Renumber Data Items. New data name is generated using printf-like `data_name_fmt` and data counter that runs from `data_name_start` with step `data_name_step`.

Configuration parameters:

- * `VString data_name_fmt`
New data name format. Should have exactly one `%d` for data name counter.
- * `int data_name_start`
Data name counter start value.
- * `int data_name_step`
Data name counter step.

9.3 Cobol Grep Transformation

This transformation searches for node that satisfies specified query. Queries are written in **Query Language (QL)**. QL manual is available separately.

- **ScvtGrep**

Find Things in Cobol code.

Configuration parameters:

- * `VString query_str`
Query string.
- * `VString query_file_name`
Name of file that contains query text.
- * `int print_query`
Print query exactly as the parser sees it.
- * `int print_src`
Print source line of the tree node that matches the query.
- * `int print_tree`
Print subtree that matches the query.
- * `int print_err_pos`
If query is incorrect, indicate error position in query.

9.4 File and Data Descriptor Extractors

- **Block ScvtCbl2Fdd**
Generate RDD and FDD files for file or data item.
- **ScvtWriteDataItemDescriptors**
Write data item descriptors to file.
- **ScvtWriteFileDescriptors**
Write file descriptors to file.
- **ScvtComputeBitLengthsOffsets**
Compute lengths and offsets for Bit (FSC) record items.

9.5 IBM printer control characters simulation

- **ScvtProcessIBMPrinterFiles**
Simulate behavior of IBM printer files control characters.

9.6 IBM Cobol To Fujitsu Cobol Transformations

- **Block ScvtIbm2Fsc**
Convert IBM Cobol dialects to Fujitsu Cobol97.
Configuration parameters:
 - * **int add_value_clause**
If True, add VALUE clauses to data items in WS that do not have it.
 - * **int comment_execs**
If True, comment out EXEC ... END-EXEC statements.
 - * **int max_report_length**
If ≥ 0 , then add RECORD CONTAINS clause to FDs with REPORT clause.
 - * **int expand_expressions**
If True, expand abbreviated combined conditions.
 - * **VString fsc_options**
FSC compiler options.
- **ScvtRemoveDataRecordsClauseFromFDwithReportClause**
Remove DATA RECORDS clause from FD with REPORT clause.
- **ScvtRemoveRecordDescrFromFDwithReportClause**
Remove record description from FD with REPORT clause.
- **ScvtWriteAfterPosngLinesToWriteAfterAdvngLines**
Convert 'WRITE ... AFTER POSITIONING x LINES' to 'WRITE ... AFTER ADVANCING 0-3 LINES'.

- **ScvtExpandAbbreviatedConditions**
Expand abbreviated conditions (UNFINISHED).

9.7 Micro Focus to Fujitsu Transformations

- **Block ScvtMf2Fsc**
Convert Micro Focus Cobol dialects to Fujitsu Cobol97.
- **ScvtAssignComputedValuesTo78s**
Assign previously computed value to 78-level constant.
- **Scvt78LevelItemsToReplaceStatements**
Turn 78-level constant into REPLACE statement.
- **ScvtPropagate78LevelItemsInPictures**
Substitute 78-level constant used in a picture by its value.
- **ScvtPropagate78LevelItems**
Propagate value of a 78-level constant to all places where it is used including pictures.
- **ScvtStringLiteralsToNumericLiteralsInBinaryComparisons**
Converted hexadecimal literal to decimal literal when compared to numeric value.
- **ScvtExitParagraphToGoTo**
Convert EXIT PARAGRAPH statement to GO TO statement.
- **ScvtExitSectionToGoTo**
Convert EXIT SECTION statement to GO TO statement.
- **ScvtRemoveByValueSpecifiersInCallParameters**
Remove BY VALUE argument specifier in CALL parameters and create auxiliary variables to prevent program logic.
- **ScvtCreatePeersForDisplayedOrAcceptedFSLSEntries**
Create WS peer for variable defined outside WS section in order to use it as DISPLAY/ACCEPT argument.
- **ScvtCreateLSPeersForProgramParametersDefinedInWSSection**
Create LINKAGE SECTION peer for data entry defined outside LINKAGE SECTION in order to use it as program parameter.
- **ScvtUponPrinterToUponSysout**
Convert DISPLAY UPON PRINTER to DISPLAY UPON SYSOUT.
- **ScvtMSStyleDisplayItemPositioningToFSCStyle**
Convert MS Cobol style of display item position specification to the commonly used COLUMN X LINE Y.

- **ScvtMSStyleDisplayErasingItemToFSCStyle**
Convert MS Cobol style of specifying erasing display item to the commonly used one.
- **ScvtMSStyleAcceptItemPositioningToFSCStyle**
Convert MS Cobol style of ACCEPT item position specification to the commonly used one.
- **ScvtMSMakeLinColExplicit**
Create explicit definitions for MS Cobol special registers LIN and COL.
- **ScvtAnsiToScreen**
Convert DISPLAY or ACCEPT item from ANSI format to screen format.
- **ScvtFitColorNumbersIntoTheRangeOf0to7**
Suppress upper bits for color number exceeding 7.
- **ScvtFindIllegalClausesInAcceptAndDisplay**
Find unsupported ACCEPT/DISPLAY features.
- **ScvtDeleteEraseClauseFromDisplayStatement**
Remove ERASE clause in DISPLAY statement.
- **ScvtRMStyleAdisFeaturesToFSCStyle**
RM Cobol: convert features of DISPLAY or ACCEPT statement.
- **ScvtUsageComp1ToUsageComp**
RM Cobol: convert USAGE COMP1 to USAGE COMP.
- **ScvtShiftLevelsOf01And77LevelDataItemsWithOccurs**
Create additional data level to prevent OCCURS clause from appearing at level 01.
- **ScvtShiftLevelsOfRedefinedDataItemsWithOccurs**
Create additional data level so that REFEDINES does not refer to data item with OCCURS clause.
- **ScvtLevelMutuallyRedefiningItems**
Make sure that level of the item being redefined is equal to the level of redefining one.
- **ScvtNumericLiteralToStingLiteralForNumericEditedValues**
Convert initial value for numeric edited data items from numeric form to string form.
- **ScvtAddOffsetsToLineAndColumnClauses**
Add numeric offsets to LINE or COLUMN clause to make it comply with the new standard.
- **ScvtDeleteNumericSignClauseInSpecialNamesParagraph**
Delete NUMERIC SIGN IS TRAILING SEPARATE clause.
- **ScvtAddNumericSignIsTrailingSeparateWhereNeeded**
Add SIGN IS TRAILING SEPARATE to data item in scope of NUMERIC SIGN IS TRAILING SEPARATE directive.

- **ScvtDeleteNotOptionalPhrase**
Delete NOT OPTIONAL phrase.
- **ScvtOrgainizationRecordSequentialToOrgainizationSequential**
Convert ORGANIZATION RECORD SEQUENTIAL to ORGANIZATION SEQUENTIAL.
- **ScvtAddLeadingZeroDigitToHexadecimalLiteralsWithOddNumbersOfDigits**
Add "0" digit at the beginning of hexadecimal literal with odd number of digits.
- **ScvtCreateIntegerPeersForIndexNamesUsedInImproperContexts**
Create integer peer for INDEX-NAME in order to use it in context where INDEX-NAME use is not allowed by the new standard.
- **ScvtAddParenthesesAroundArgumentsInClassCheckConditions**
Add parentheses around arguments of class-check condition.
- **ScvtH99AABBCCIntegerLiteralsToRegularIntegerLiterals**
Convert hexadecimal integer value represented by string literal to numeric literal.
- **ScvtRemoveLockModeWithRollBackClause**
Remove WITH ROLLBACK clause.
- **ScvtLockModeManualToLockModeAutomaticForSequentialFiles**
Convert MANUAL lock mode to AUTOMATIC lock mode for sequential file.
- **ScvtConvertLockModeClauseForRelativeAndIndexedFiles**
Convert LOCK MODE for relative/indexed file.
- **ScvtReplaceDeleteFileStatementByRuntimeSupportFunctionCall**
Convert DELETE FILE statement to CALL statement.
- **ScvtValueOfFileIdToCorrespondingAssignClause**
Convert VALUE OF FILE ID clause to ASSIGN clause.
- **ScvtConvertAssignToClauseForSortMergeFiles**
Convert ASSIGN clause for SORT/MERGE file.
- **ScvtDeleteExternalDynamicSpecifiersInAssignClause**
Delete EXTERNAL/DYNAMIC specification for MF files.
- **ScvtLineAdvancingFileToRegularFile**
Convert LINE ADVANCING file to regular file.
- **ScvtUnsupportedDeviceNamesToTheCorrespondingRegularDeviceNames**
Converted ASSIGN TO 'prn:/'lpt1:' to ASSIGN TO PRINTER.
- **ScvtFindUnsupportedDeviceNamesInAssignToClauses**
Find unsupported device name in ASSIGN TO clauses.

- ScvtRandomFileToDiskFile
Convert RANDOM file to DISK file.
- ScvtSortStatusVariableToSortStatusSpecialRegister
Replace user-defined variable storing sort status by SORT-STATUS special register.
- ScvtRegisterIOStatements
Resister all IO statements for use on the next passes.
- ScvtAddIOStatusConversionCallsInExceptionBranches
Add IO Status conversion call to exception branches of IO statement.
- ScvtAddIOStatusConversionCallsInMainBranch
Add IO Status conversion call after IO statement.
- ScvtAddEndStatementClausesToAllIOStatements
Add END-STATEMENT delimiter to IO statement.
- ScvtDeleteReadWithWait
Delete WITH WAIT clause of READ statement.
- ScvtCommitAndRollbackStatementsToRuntimeCalls
Convert COMMIT or ROLLBACK statement to CALL statement.
- ScvtAddExplicitExternalSpecificationToRelativeKeysForExternalFiles
Specify explicitly that relative key for external file is external.
- ScvtDeleteFDValueItems
Delete VALUE OF clause in file descriptor entry.
- ScvtAddConsoleIsCRTInSpecialNamesParagraph
Add CONSOLE IS CRT phrase into SPECIAL NAMES paragraph.
- ScvtCallGivingToCallReturning
Convert CALL ... GIVING to CALL ... RETURNING.
- ScvtChainingToUsing
Convert PROCEDURE DIVISION CHAINING to PROCEDURE DIVISION USING.
- ScvtChainStatementToCallStatement
Convert CHAIN statement to CALL statement.
- ScvtDeleteCallConventionDefinitionsInSpecialNames
Delete CALL CONVENTION entries in SPECIAL NAMES paragraph.
- ScvtMFStyleCallConventionSpecificationToFSCStyle
Convnet API call convention mnemonic to corresponding WITH LINKAGE clause.
- ScvtCall_XAA_To_Call_CBL_SYS_AA
Replace CALL X'AA' by CALL CBL_SYS_AA.

- ScvtRestoreCorrespondenceBetweenActualAndShownParameterTypes1
Set shown parameter type to the actual parameter type - case 1.
- ScvtRestoreCorrespondenceBetweenActualAndShownParameterTypes2
Set shown parameter type to the actual parameter type - case 2.
- ScvtByRefToByContentForLiteralsInCallParameters
Convert BY REF <literal> to BY CONTENT <literal> in CALL parameters.
- ScvtPrepareByValueParametersDefinedInLinkageSection
Auxiliary conversion preparing the ground for the next one.
- ScvtFindSwitch8
Find SWITCH-8.
- ScvtSetToStatementToMoveStatement
Convert SET ... TO statement for numeric data items to MOVE ... TO statement.
- ScvtSetUpDownByIntegerStatementToComputeStatement
Convert SET ... UP/DOWN statement for numeric data items to COMPUTE statement.
- ScvtAcceptFromCommandLineToRuntimeCall
Convert ACCEPT FROM COMMAND-LINE statement to CALL statement.
- ScvtAcceptFromEscapeKeyToMoveStatement
Convert ACCEPT FROM ESCAPE KEY statement to MOVE statement.
- ScvtDisplayUponCommandLineToRuntimeCall
Convert DISPLAY UPON COMMAND-LINE statement to CALL statement.
- ScvtArgOrEnvValOrNumberFunctionNamesToMnemonicNames
Convert specific function name to mnemonic name.
- ScvtNextSentenceStatementToGoToStatementInExceptionHandler
Delete NEXT SENTENCE statement in exception branch or replace it by GO TO statement.
- ScvtMoveWithIncompatibleParametersToTheSequenceOfMoveStatements
Convert MOVE statement with incompatible operand types to sequence of two MOVE statements using intermediate variable.
- ScvtRemoveValueWhenSetToFalseClause
Delete value WHEN SET TO FALSE clause of data item.
- ScvtSetToFalseStatementToMoveStatement
Convert SET ... TO FALSE statement to MOVE statement using information provided in WHEN SET TO FALSE clause of data item.
- ScvtExpandPartialExpressionsInWhenPhrases
Expand partial expression in WHEN phrase adding missing parts to it.

- **ScvtReplaceEvaluationSubjectsByTheTRUEValue**
Replace evaluation subject by the value "TRUE".
- **ScvtSpecialNamesPrinterToSpecialNamesSysout**
Use SYSOUT insted of PRINTER in SPECIAL NAMES.
- **ScvtDisplayAtGroupItemToDisplayAtFourDigitNumeric**
When a group item is used as DISPLAY AT argument redefine it with a numeric item of proper length.
- **ScvtRemoveAcceptUpperLower**
Remove UPPER/LOWER clauses in ACCEPT statement.
- **ScvtSingleDoubleQuotesWorkaround**
Implements workaround that helps FSC to work with string literals in some casess.

9.8 Loaded Transformation Runner

- **ScvtRunTrans**
Run Named Transformation.

9.9 Finder Transformations

- **ScvtFindObjectDefinitionsByName**
Find definitions for objects whose names start with a given **name**. Case-insensitive.
Configuration parameters:
 - * **VString name**
First characters of the name of te object we are looking for.
- **ScvtFindFileDescriptors**
Find file descriptors (FDs).
- **ScvtFindCallStatements**
Find CALL and CHAIN statements.

9.10 IBM Cobol Dialects Normalization Transformations

- **Block ScvtIbm2FscBase**
Convert IBM Cobol dialects to Fujitsu Cobol97 – Base.
Configuration parameters:
 - * **int max_report_length**
If ≥ 0 , then add RECORD CONATINS clause to FDs with REPORT clause.

- * `int comment_execs`
If True, comment out EXEC ... END-EXEC statements.
 - * `VString fsc_options`
FSC compiler options.
- `ScvtCurrentDateToFunc`
Convert use of CURRENT-DATE special register (OSVS) to use of ANSI-85 date functions.
- `ScvtTimeOfDayToFunc`
Convert use of TIME-OF-DAY special register (OSVS) to use of ANSI-85 time functions.
- `ScvtBuildSysDateRecord`
Build FSC-SYS data record; FSC-CUR-DATE and FSC-CUR-TIME data items if they are not defined yet.
- `ScvtCurrentDateToFuncWithContext`
Convert CURRENT-DATE and TIME-OF-DAY special registers to calls to ANSI-85 functions.
This transformation has internal variable that are initialized at the start of the Cobol Program.
- `ScvtTransformStmtToInspectStmt`
Convert OSVS Transform statement to ANSI Inspect Converting statement.
- `ScvtNormalizeIndexPlusMinusPositiveInteger`
Convert TABLE (index -/+ +x) to TABLE (index -/+ x).
- `ScvtNormalizeIndexPlusMinusNegativeInteger`
Convert TABLE (index -/+ -x) to TABLE (index +/- x).
- `ScvtAddRecordContainsClauseToFDsWithReport`
Add RECORD CONTAINS `max_report_length` clause to FDs that have REPORT clause and have no RECORD CONTAINS clause.
Configuration parameters:
 - * `int max_report_length`
Record length to put into added RECORD CONTAINS clause.
- `ScvtExhibitUnnamedStmtToDisplayStmt`
Convert EXHIBIT (no NAMED) statement to DISPLAY statement.
- `ScvtMoveUnnamedExhibitItemToDisplayItem`
Move Unnamed EXHIBIT item to DISPLAY item.
- `ScvtExhibitNamedStmtToDisplayStmt`
Convert EXHIBIT NAMED statement to DISPLAY statement.
- `ScvtMoveNamedExhibitItemToDisplayItem`
Move NAMED EXHIBIT item to DISPLAY item.

- **ScvtWriteAdvMnemLinesToWriteAdvOneLine**
Convert 'WRITE file BEFORE/AFTER ADVANCING mnemonic-name LINES' to 'WRITE file BEFORE/AFTER ADVANCING 1 LINES'.
- **ScvtReportLineNextPageToLineOneNextPage**
Report Writer: Change LINE NEXT PAGE to LINE 1 NEXT PAGE.
- **ScvtIfOtherwiseTolElse**
Convert 'IF ... THEN ... OTHEWISE' to 'IF ... THEN ... ELSE ...'.
- **ScvtRemoveChannelDeclsFromSpecialNames**
Remove channel C01-C12 and S01-S05 declarations from SPECIAL-NAMES paragraph.
- **ScvtRemoveRecordingModeVarIfWithRecordsVarying**
Remove 'RECORDING MODE IS V' clause if it is specified together with 'RECORD IS VARYING IN SIZE' clause.
- **ScvtPropagateSyncClauseDownToElementary**
Propagate SYNC clause from composite data item declaration down to the elementary levels and remove it from the composite level.
- **ScvtAddSyncClauseToElementaryItem**
Add SYNC clause to elementary data items.
- **ScvtCommentOutExecStmtInProc**
Comment out EXEC statement in Procedure division.
- **ScvtCommentOutExecStmtInData**
Comment out EXEC statement in Data division.

9.11 Visual Macro

- **ScvtVMacro**
Visual Macro.

9.12 SQL Transformations

- **Block ScvtGeneralSqlToFscSql**
Convert general-case SQL to Fujitsu Cobol SQL.
- **ScvtRemoveSqlSysVarsUserDefs**
Remove SQL system variables SQLSTATE and SQLCODE so that target system can define them to its own spec.
- **ScvtFixHostSqlHostVarsForFsc**
Find DECL_DD_ENTRY that is used as a SQL host variable and that is not 01-level variable or otherwise does not satisfy FSC criteria for host variable. Create 01-level peer for such variable.

- **ScvtRemoveSqlIncludes**
Remove SQL INCLUDE SQLCA and other such includes.
- **ScvtAddSqlStateCodeDecls**
Add explicit SQLCODE and SQLSTATE declarations if they are missing.
- **ScvtDeleteAllSqlBeginEndBrackets**
Delete all EXEC SQL BEGIN/END DECLARATION SECTION statements.
- **ScvtAddTrueSqlBeginEndBrackets**
Add SQL BEGIN/END DECLARE SECTION brackets around true host variable declarations.
- **ScvtDeleteExcessiveSqlBeginEndBrackets**
Delete excessive EXEC SQL BEGIN/END DECLARATION SECTION statements.
- **ScvtRemoveDeclareTable**
Remove SQL DECLARE TABLE statement that is mostly FYI.

9.13 Standard Transformations

- **ScvtBlock**
Applies the referenced transformations to a given subtree.
Transformation operations have the following meaning for **ScvtBlock**:

```
* SctErrCode ApplyAll(SctExpr *e, int transform, SctTokenList &msgs,
  SctExpr **pe_after = NULL)
```

 We override **SctTransform::ApplyAll()** to provide custom behaviour: If flag **by_trans_first** is True, then for every referenced transformation **t** apply **t** to all nodes in subtree **e**. If flag **by_trans_first** is False, then for every node **ee** in subtree **e** apply all referenced transformations to **ee**.
- **ScvtInitAndApplyRefdToCobolProgram**
Initialize and apply all referenced transformations to Cobol program.
That is, for every Cobol program initialize all referenced transformations and then call **ApplyAll()** of each referenced transformation on the program.
- **Block ScvtNormalizeCode**
Normalize code by bringing it closer to ANSI-85 syntax Applicable to all dialects.
- **Block ScvtNormalizeMicroFocus**
Normalize Micro Focus code by bringing it closer to ANSI-85 syntax.
- **ScvtRenameKeyword**
Rename words that are keywords in dialect **dial_to** by adding suffix **suffix** to them.
Configuration parameters:

- * `int dial_to`
Dialect to which we are renaming.
 - * `VString suffix`
Suffix added to the word when renaming.
- **ScvtRenameNamesWithDigitsOnly**
Add prefix Q- to names that consist of digits only.
Such names are allowed in MicroFocus but not in the ANSI standard.
- **ScvtAddCompilerSettingsLineAtTheTop**
Add specified compiler `settings` comment-like line at the beginning of file.
Configuration parameters:
 - * `VString settings`
Settings comment-like line to be added to the beginning of file.
- **ScvtRemoveNullNodesInListNodes**
Remove Null nodes in list nodes.
- **ScvtRemoveEmptyListNodes**
Remove list expression nodes that have no children.
- **ScvtAddStdDivisionSectionHeaders**
Add standard division and section headers where needed.
In Micro Focus and some other dialects you may skip certain division and section headers required by standard. This transformation restores them.
- **ScvtAddProgramName**
Set Program Name to specified parameter if it is empty.
Configuration parameters:
 - * `VString def_prog_name`
Default program name that is given to a program that has no name (possible in MF).
- **ScvtNormalizeClauses**
Turn non-standard Micro Focus rendition of certain clauses into their ANSI-85 version.
- **ScvtAddSuffixAndQuotesToCopy**
Convert `COPY file` and `COPY "file"` to `COPY "file.ext"`.
We need this because some compilers (FSC) cannot handle copylib file that has no explicit extension.
- **ScvtIncludeToCopy**
Convert `++INCLUDE` and `-INC` statements to regular `COPY` statement.
- **ScvtNormalizeComments**
Convert non-standard comments to standard column 7 comments.
- **ScvtNoteToComment**
Turn `NOTE` statement into comments.

- **ScvtShortenLongNames**
Rename names that are longer than 30 characters to shorter but still unique names. Achieve uniqueness by adding numeric suffixes.
- **ScvtRemoveLabelRecordsClause**
Remove LABEL RECORDS clause that is ignored by most compilers anyway.
- **ScvtRemoveExtraParensInEvaluateStmt**
Remove parentheses around objects/subject in ALSO phrase of EVALUATE stmt.
- **ScvtAddSpacesAroundOperators**
Add spaces around arithmetic and relational operations where needed.
- **ScvtNormalizeNonnumLiterals**
Normalize non-numeric literals: upper-case starting 'X', fix area B in continuation.
- **ScvtAddEndBracketStatements**
Add ANSI-required/all closing END-IF, END-SEARCH, END-EVALUATE, END-PERFORM statements. Adding only required closing statements brings code in conformance with ANSI-85.
Configuration parameters:
 - * **int all**
True: add closing statement to all places where possible. False: add closing statements only where required by ANSI-85 and not required by MF and the like.
- **ScvtNormalizeDataItemLevelNumbers**
Normalize level numbers in data item descriptions so that level numbers for all subordinates of a data item are the same.
- **ScvtRemoveValueClauseFromDataDeclsInFileLinkageSections**
Remove VALUE clause from data item declarations in FILE and LINKAGE sections.
- **ScvtAddInitialValuesToDataEntryWithNoValue**
Add VALUE clause to data items that have no initial value specified.
- **ScvtDeleteSegmentNumbersFromSectionHeaders**
Delete segment-number from SECTION header.
- **ScvtUsageComp0toBinary**
USAGE COMP-0 to USAGE BINARY.
- **ScvtUsageComp4toBinary**
USAGE COMP-4 to USAGE BINARY.
- **ScvtUsageCompXtoBinary**
USAGE COMP-X to USAGE BINARY.
- **ScvtBinaryPicXsToPic9s**
Binary data items with PIC X(n) to PIC 9(approx n*2).

- **ScvtUsageBinaryToDisplayXs**
Convert USAGE BINARY PIC 9(n) to USAGE DISPLAY PIC X(byte-length).
- **ScvtDeleteSplitKeyDefinition**
Delete definition of a split key.
- **ScvtSplitKeyToSplitKeyComponentList**
Replace user-defined split key by the list of its components.
- **ScvtMSStyleLockingClauseToTheCorrespondingFSClockModeClause**
MS Cobol: convert LOCKING IS clause.
- **ScvtSplitMultiitemDisplayStatements**
Split DISPLAY statement with several items into sequence of single-item DISPLAY statements.
- **ScvtSplitStatementsWithItemList**
Split a statement that contains a list of similar clauses to several statements each having only single clause..
Configuration parameters:
 - * **SctOper stmt_oper**
Particular kind of statements that we are splitting.
 - * **int stmt_arg_to_split**
Number of argument containing the list to split.

9.14 Utility Transformations

- **ScvtGoToEnclosingStatement**
Go to Enclosing Statement.
- **ScvtIsSpecialRegister**
Applies if the current node is Special Register.
- **ScvtResolveNamedObjUse**
Resolve Definition for NamedObjectUse node.
- **ScvtFlattenList**
Flatten two-level list with the same operation at both levels to one level.
- **ScvtNormalizeConstant**
Normalize Constant: replace its image with its normalized value.
- **ScvtTurnExprIntoComments**
Turn expression into comments that are attached to the specified statement that replaces this expression.

9.15 Year 2000 Fix Transformations

- ScvtY2kWindowFix

Fix Y2K problem by applying Windowing to selected data items. Data items are selected either by their PICTure or from the list of data items stored in text file.

Configuration parameters:

- * **VString date_pic**

If not null, PICTURE format of data items to which Y2K windowing should be applied.

- * **VString date_file_name**

If not null, file name for file which contains list of data items to which Y2K windowing should be applied.

Chapter 10

Cobol Program Tree: Reference

This Chapter describes Cobol Program Tree operations.

Format of the Description Table. For every Cobol operation described here we provide a table below.

In the first row of this table we have the following columns:

- Operation code.
- Description of the operation.
- Code generation table entry for this operation. The entry may be incomplete. See file `cobol.tdl` for complete code generation table entries.

Rows from the 2nd to the last correspond to the operation arguments (children), one row per argument. Each argument row contains:

- Argument index (from 1 for the first argument to Number of arguments).

If after the numeric index there is a colon and a name, this name is a named index of the argument.

Named indices are good to have when there are many positional arguments at one node and numeric indices are hard to memorize or are not intuitive enough.

- Description of the argument.
- Expression operations that are allowed at this argument of this operation.

Template for the table looks like this:

| Operation Code | Operation Description | CODE GEN TABLE ENTRY \$1. |
|----------------------|-----------------------|-------------------------------------|
| \$ArgNo : INDEX_NAME | Argument Description | ALLOWED_OP1 ALLOWED_OP2 NULL |

Describing Allowed Operations at the Argument Nodes. We use a modification of Backus-Naur Form (BNF) to describe what operations are allowed to appear at the argument nodes of a given operation.

The notation used is the following:

- `OPER_1 | OPER_2 | OPER_3 | ...` means that `OPER_1` or `OPER_2` or `OPER_3` and so on may appear.
- `LIST_OPER (OPER_1 | OPER_2 | OPER_3 | ...)` means that a list of 0, 1, 2 or more arguments with operations `OPER_1`, `OPER_2`, `OPER_3` and so on may appear.

List of 0 elements is just a `ENULL` pointer, no children nodes.

List of 1 or more elements is a list operation (there are many list operations: `EX_LIST`, `EX_LIST_PARAMS`, `EX_LIST_STMT` and so on) with list elements attached as children to it.

Named Combinations (Non-Terminals). There are some combinations of allowed operations that are fairly complex. They are given a name so that we do not have to repeat their definition in all the places where they are allowed to appear.

These combinations appear in all lowercase letters and they represent common Cobol constructs such as expression, qualified name, identifier, etc.

Definition of a named combination (non-terminal) is marked with a • bullet.

Source File and Program. At the top of the tree there is a `COBOL_FILE` node. It corresponds to the Compilation Unit(s) which were parsed from a single source Cobol file.

Please note that one source file may contain several Cobol programs according to ANSI-85 standard.

10.1 Generic Operations

| | | |
|-----------|-------------------|----|
| NULL_NODE | Null Node | |
| OP_NONE | No Such Operation | |
| EX_LIST | List | @& |

10.2 Cobol File

| | | |
|---------------------------|----------------------|---|
| COBOL_FILE | Cobol File | \r\$1{\$2=\$2} |
| \$1: FILE_CONTAINED_PROGS | Program List | COBOL_PROGRAM_LIST (COBOL_PROGRAM) |
| \$2: FILE_EOF | End Of File | END_COBOL_FILE |
| \$3: FILE_EXTERNAL_DEFS | External Definitions | DECL_DD_LIST (DECL_DD) NULL |
| \$4: FILE_SYMBOL_TABLE | Symbol Table | SYMBOL_TABLE_ELT_LIST (SYMBOL_TABLE_ELT) |

| | | |
|----------------|-------------------|--|
| END_COBOL_FILE | End Of Cobol File | |
|----------------|-------------------|--|

| | | |
|--------------------|--------------------|--------|
| COBOL_PROGRAM_LIST | Cobol Program List | \r@ \n |
|--------------------|--------------------|--------|

| | | |
|---------------------------|-------------------------|--|
| COBOL_PROGRAM | Cobol Program | \$1{\$2?\$2}{\$3?\$3}{\$4?\n\$4}{\$5?\n\$5}{\$6?\$6} |
| \$1: PROG_IDENT_DIV | Identification Division | DIVISION_IDENTIFICATION |
| \$2: PROG_ENVIR_DIV | Environment Division | DIVISION_ENVIRONMENT NULL |
| \$3: PROG_DATA_DIV | Data Division | DIVISION_DATA NULL |
| \$4: PROG_PROC_DIV | Procedure Division | DIVISION_PROCEDURE |
| \$5: PROG_CONTAINED_PROGS | Contained Programs | COBOL_PROGRAM_LIST (COBOL_PROGRAM) NULL |
| \$6: PROG_END_PROGRAM | End Program | END_COBOL_PROGRAM NULL |
| \$7: PROG_SYMBOL_TABLE | Symbol Table | SYMBOL_TABLE_ELT_LIST (SYMBOL_TABLE_ELT) |

| | | |
|--------------------------------|--------------------|--------------------|
| END_COBOL_PROGRAM | End Program header | END PROGRAM \$1[.] |
| \$1: END_COBOL_PROGRAM_NAME | Program Name | NAMED_OBJ_USE |

| | | |
|-----------------------|---------------------------|--|
| SYMBOL_TABLE_ELT_LIST | Symbol Table Element List | |
|-----------------------|---------------------------|--|

| | | |
|------------------|----------------------|--|
| SYMBOL_TABLE_ELT | Symbol Table Element | |
|------------------|----------------------|--|

10.3 IDENTIFICATION Division

| | | |
|--------------------------------|--------------------------------|--|
| DIVISION_IDENTIFICATION | Identification Division | \$1{\$2?\n\$2} |
| \$1: DI_HEADER | Division Header | DIVISION_IDENTIFICATION_HEADER NULL |
| \$2: DI_PROGRAM_ID | Program-Id Paragraph | PARA_PROGRAM_ID NULL |
| DIVISION_IDENTIFICATION_HEADER | Identification Division Header | IDENTIFICATION DIVISION. |
| PARA_PROGRAM_ID | Program-Id Paragraph | PROGRAM-ID.{ \$1? \$1 } { \$2? \$2 } [.] |
| \$1: PPID_NAME | Program Name Defined | NAMED_OBJ_DEF |
| \$2: PPID_ATTR | Program Attributes | EX_PROG_COMMON EX_PROG_INITIAL EX_PROG_COMMON_INITIAL NULL |
| EX_PROG_COMMON | Common Attr | IS COMMON PROGRAM |
| EX_PROG_INITIAL | Initial Attr | IS INITIAL PROGRAM |
| EX_PROG_COMMON_INITIAL | Common Initial Attrs | IS COMMON INITIAL PROGRAM |

10.4 ENVIRONMENT Division

| | | |
|-----------------------------|-----------------------------|---|
| DIVISION_ENVIRONMENT | Environment Division | { \$1? \n \$1 } { \$2? \$2 } { \$3? \$3 } |
| \$1: DE_HEADER | Header | DIVISION_ENVIRONMENT_HEADER NULL |
| \$2: DE_SEC_CONFIG | Configuration Section | SECTION_CONFIGURATION NULL |
| \$3: DE_SEC_IO | Input-Output Section | SECTION_INPUT_OUTPUT NULL |
| DIVISION_ENVIRONMENT_HEADER | Environment Division Header | ENVIRONMENT DIVISION. |

10.4.1 CONFIGURATION Section

| | | |
|-------------------------|-----------------------------|---|
| SECTION_CONFIGURATION | Configuration Section | { \$1? \n \$1 } { \$2? \n \$2 } { \$3? \n \$3 } { \$4? \n \$4 } |
| \$1: CS_HEADER | Header | SECTION_CONFIGURATION_HEADER NULL |
| \$2: CS_SOURCE_COMPUTER | Source Computer | PARA_SOURCE_COMPUTER NULL |
| \$3: CS_OBJECT_COMPUTER | Object Computer | PARA_OBJECT_COMPUTER NULL |
| \$4: CS_SPEC_NAMES | Special Names | PARA_SPECIAL_NAMES NULL |
| \$5: CS_FIG_CONSTANTS | Figurative Constants (Wang) | PARA_FIG_CONSTS NULL |
| \$6: CS_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|------------------------------|-----------------------|------------------------|
| SECTION_CONFIGURATION_HEADER | Configuration Section | CONFIGURATION SECTION. |
|------------------------------|-----------------------|------------------------|

SOURCE-COMPUTER paragraph

| | | |
|------------------------|---------------------------|---|
| PARA_SOURCE_COMPUTER | Source-Computer Paragraph | SOURCE-COMPUTER.{ \$1? \$1 } { \$2? \$2 } [.] |
| \$1: PSC_COMPUTER_NAME | Source Computer Name | CHAR_STRING NULL |
| \$2: PSC_DEBUG_MODE | Debug Mode | EX_CC_DEBUGGING_MODE NULL |

| | | |
|----------------------|---------------------|---------------------|
| EX_CC_DEBUGGING_MODE | With Debugging Mode | WITH DEBUGGING MODE |
|----------------------|---------------------|---------------------|

OBJECT-COMPUTER paragraph

| | | |
|------------------------|---------------------------|--|
| PARA_OBJECT_COMPUTER | Object-Computer Paragraph | OBJECT-COMPUTER. \$1 { \$2? \$2 } [.] |
| \$1: POC_COMPUTER_NAME | Object Computer Name | CHAR_STRING |
| \$2: POC_CLAUSES | Clauses | EX_CC_CLAUSE_LIST (EX_CC_MEMORY_SIZE EX_CC_COLLATING_SEQUENCE EX_CC_SEGMENT_LIMIT) NULL |

| | | |
|-------------------|----------------------------------|----|
| EX_CC_CLAUSE_LIST | Object-Computer Para Clause List | @& |
|-------------------|----------------------------------|----|

| | | |
|-------------------|--------------------|--|
| EX_CC_MEMORY_SIZE | Memory Size Clause | MEMORY \$1 { \$2= \$2 } |
| \$1: CCMS_SIZE | Memory Size | INT_NUM_CONST |
| \$2: CCMS_UNITS | Units | EX_CC_WORDS EX_CC_CHARACTERS EX_CC_MODULES |

| | | |
|-------------|-----------------|-------|
| EX_CC_WORDS | Words Size Unit | WORDS |
|-------------|-----------------|-------|

| | | |
|------------------|----------------------|------------|
| EX_CC_CHARACTERS | Characters Size Unit | CHARACTERS |
|------------------|----------------------|------------|

| | | |
|---------------|-------------------|---------|
| EX_CC_MODULES | Modules Size Unit | MODULES |
|---------------|-------------------|---------|

| | | |
|--|-----------------------------------|-----------------------------------|
| EX_CC_COLLATING_SEQUENCE | Program Collating Sequence Clause | PROGRAM COLLATING SEQUENCE IS \$1 |
| \$1: EX_CC_COLLATING_SEQUENCE__ALPHABET | Alphabet Name | NAMED_OBJ_USE |

| | | |
|-----------------------------------|----------------------|----------------------|
| EX_CC_SEGMENT_LIMIT | Segment Limit Clause | SEGMENT-LIMIT IS \$1 |
| \$1: EX_CC_SEGMENT_LIMIT_LIMIT | Limit Value | INT_NUM_CONST |

SPECIAL-NAMES paragraph

| | | |
|----------------------------|---------------------------|--|
| PARA_SPECIAL_NAMES | Special-Names Paragarph | SPECIAL-NAMES. \$1{\$2?\n\t\$2\b}{\${3?\n |
| \$1: SN_NOT_USED | not-used | NULL |
| \$2: SN_FUNC_SWITCH | Mnemonic/Switch | SNX_FUNC_SWITCH_LIST (SNX_FUNC_SWITCH) NULL |
| \$3: SN_ALPHABET | Alphabet | SNX_ALPHABET_LIST (SNX_ALPHABET) NULL |
| \$4: SN_SYMB_CHARS | Symbolic Characters | SNX_SYMB_CHARS_LIST (SNX_SYMB_CHARS) NULL |
| \$5: SN_CLASS | Class | SNX_CLASS_LIST (SNX_CLASS) NULL |
| \$6: SN_CURRENCY_SIGN | Currency Sign | SNX_CURRENCY_SIGN NULL |
| \$7: SN_DECIMAL_POINT | Decimal Point | SNX_DECIMAL_POINT NULL |
| \$8: SN_NUMERIC_SIGN | Numeric Sign | SNX_NUMERIC_SIGN_IS_TS NULL |
| \$9: SN_CONSOLE | Console Is Crt | SNX_CONSOLE_IS_CRT NULL |
| \$10: SN_CALL_CONVENTION | Call-Convention | SNX_CALL_CONVENTION NULL |
| \$11: SN_CURSOR | Cursor | SNX_CURSOR NULL |
| \$12: SN_CRT_STATUS | Crt Status | SNX_CRT_STATUS NULL |
| \$13: SN_SYMBOLIC_CONSTANT | Symbolic Constant | SNX_SYMBOLIC_CONSTANT NULL |
| \$14: SN_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|----------------------|-----------------------------|-----|
| SNX_FUNC_SWITCH_LIST | Mnemonic/Switch Clause List | @\n |
|----------------------|-----------------------------|-----|

| | | |
|-------------------------------------|------------------------|--|
| SNX_FUNC_SWITCH | Mnemonic/Switch Clause | \$1{\$2? \$2}{\$3? \$3}{\$4? \$4} |
| \$1: SNX_FUNC_SWITCH__SW_IOF | Switch/IOfunction | SW_SWITCH_0 SW_SWITCH_1 SW_SWITCH_2 SW_SWITCH_3 SW_SWITCH_4 SW_SWITCH_5 SW_SWITCH_6 SW_SWITCH_7 SW_SWITCH_8 SW_UPSI_0 SW_UPSI_1 SW_UPSI_2 SW_UPSI_3 SW_UPSI_4 SW_UPSI_5 SW_UPSI_6 SW_UPSI_7 io-function |
| \$2: SNX_FUNC_SWITCH__MNEM | Is Mnemonic Phrase | SNX_IS NULL |
| \$3: SNX_FUNC_SWITCH__ON_STATUS | On Status | SNX_ON_STATUS NULL |
| \$4: SNX_FUNC_SWITCH__OFF_STATUS | Off Status | SNX_OFF_STATUS NULL |

| | | |
|----------------------|--------------------|---------------|
| SNX_IS | Is Mnemonic Phrase | IS \$1 |
| \$1: SNX_IS__DEFINED | Mnemonic Defined | NAMED_OBJ_DEF |

| | | |
|-----------------------------|-------------------|------------------|
| SNX_ON_STATUS | On Status Is | ON STATUS IS \$1 |
| \$1: SNX_ON_STATUS__DEFINED | On Status Defined | NAMED_OBJ_DEF |

| | | |
|---------------------------------|--------------------|-------------------|
| SNX_OFF_STATUS | Off Status Is | OFF STATUS IS \$1 |
| \$1: SNX_OFF_STATUS__DEFINED | Off Status Defined | NAMED_OBJ_DEF |

| | | |
|-------------------|----------------------|-----|
| SNX_ALPHABET_LIST | Alphabet Clause List | @\n |
|-------------------|----------------------|-----|

| | | |
|-------------------|------------------|---|
| SNX_ALPHABET | Alphabet Clause | ALPHABET \$1 IS{\$2= \$2} |
| \$1: SNAL_DEFINED | Alphabet Defined | NAMED_OBJ_DEF |
| \$2: SNAL_VALUE | Defined As | SNX_STANDARD_1 SNX_STANDARD_2 SNX_NATIVE SNX_EBCDIC SNX_ALPHA_RANGE_LIST (EX_ALPH_LITER_RANGE) |

| | | |
|----------------------------|---------------|-----------------------------|
| EX_ALPH_LITER_RANGE | Literal Range | \$1{\$2?%THRU \$2} |
| \$1: ALPH_LITER_RANGE_FROM | From | EX_ALPH_ALSO_LIST (literal) |
| \$2: ALPH_LITER_RANGE_TO | To | literal NULL |

| | | |
|-------------------|---------------------------|--------|
| EX_ALPH_ALSO_LIST | Alphabet Also Phrase List | @%ALSO |
|-------------------|---------------------------|--------|

| | | |
|----------------|-----------------------------|------------|
| SNX_STANDARD_1 | Standard-1 (Ascii) Alphabet | STANDARD-1 |
|----------------|-----------------------------|------------|

| | | |
|----------------|---------------------|------------|
| SNX_STANDARD_2 | Standard-2 Alphabet | STANDARD-2 |
|----------------|---------------------|------------|

| | | |
|------------|-----------------|--------|
| SNX_NATIVE | Native Alphabet | NATIVE |
|------------|-----------------|--------|

| | | |
|------------|-----------------|--------|
| SNX_EBCDIC | Ebcdic Alphabet | EBCDIC |
|------------|-----------------|--------|

| | | |
|----------------------|--------------------|----|
| SNX_ALPHA_RANGE_LIST | Literal Range List | @& |
|----------------------|--------------------|----|

| | | |
|---------------------|---------------------------------|-----|
| SNX_SYMB_CHARS_LIST | Symbolic Characters Clause List | @\n |
|---------------------|---------------------------------|-----|

| | | |
|-------------------|----------------------------|---|
| SNX_SYMB_CHARS | Symbolic Characters Clause | SYMBOLIC CHARACTERS \$1{\$2= ARE \$2}{-\$ |
| \$1: SNSC_DEFINED | Symb Chars Defined | EX_LIST (NAMED_OBJ_DEF) |
| \$2: SNSC_VALUES | Symb Chars Values | EX_LIST (INT_NUM_CONST) |
| \$3: SNSC_IN_ALPH | In what alphabet | SNX_SYMB_CHARS_IN NULL |

| | | |
|-------------------------------------|--------------------|---------------|
| SNX_SYMB_CHARS_IN | In Alphabet phrase | IN \$1 |
| \$1: SNX_SYMB_CHARS_IN__ALPHABET | Alphabet | NAMED_OBJ_USE |

| | | |
|----------------|-------------------|-----|
| SNX_CLASS_LIST | Class Clause List | @\n |
|----------------|-------------------|-----|

| | | |
|--------------------|---------------|---|
| SNX_CLASS | Class Clause | CLASS \$1{\$2= IS \$2} |
| \$1: SNCLS_DEFINED | Class Defined | NAMED_OBJ_DEF |
| \$2: SNCLS_VALUE | Class Value | SNX_ALPHA_RANGE_LIST (EX_ALPH_LITERAL_RANGE) |

| | | |
|---------------------------------|----------------------|----------------------|
| SNX_CURRENCY_SIGN | Currency Sign Clause | CURRENCY SIGN IS \$1 |
| \$1: SNX_CURRENCY_SIGN__CHAR | Character | STR_CONST |

| | | |
|-------------------|-------------------------------|------------------------|
| SNX_DECIMAL_POINT | Decimal Point Is Comma Clause | DECIMAL-POINT IS COMMA |
|-------------------|-------------------------------|------------------------|

| | | |
|------------------------|----------------------------------|-----------------------------------|
| SNX_NUMERIC_SIGN_IS_TS | Numeric Sign Is Trl. Sep. Clause | NUMERIC SIGN IS TRAILING SEPARATE |
|------------------------|----------------------------------|-----------------------------------|

| | | |
|--------------------|----------------|----------------|
| SNX_CONSOLE_IS_CRT | Console Is Crt | CONSOLE IS CRT |
|--------------------|----------------|----------------|

| | | |
|--------------------|-----------------------------|-----|
| SNX_CALL_CONV_LIST | Call Convention Clause List | @\n |
|--------------------|-----------------------------|-----|

| | | |
|---------------------|------------------------|----------------------------------|
| SNX_CALL_CONVENTION | Call Convention Clause | CALL-CONVENTION \$1{\$2= IS \$2} |
| \$1: SNCC_NUMBER | Number | INT_NUM_CONST |
| \$2: SNCC_DEFINED | Defined | NAMED_OBJ_DEF |

| | | |
|---------------------------|---------------|---------------|
| SNX_CURSOR | Cursor Clause | CURSOR IS \$1 |
| \$1: SNX_CURSOR__DATAITEM | Data Item | NAMED_OBJ_USE |

| | | |
|----------------------------------|----------------------|-------------------|
| SNX_CRT_STATUS | Crt Status Is Clause | CRT STATUS IS \$1 |
| \$1: SNX_CRT_STATUS__DATAITEM | Data -Item | NAMED_OBJ_USE |

| | | |
|----------------------------|-------------------------------|-----|
| SNX_SYMBOLIC_CONSTANT_LIST | Symbolic Constant Clause List | @\n |
|----------------------------|-------------------------------|-----|

| | | |
|-------------------------------------|--------------------------|---|
| SNX_SYMBOLIC_CONSTANT | Symbolic Constant Clause | SYMBOLIC CONSTANT\r{\$1?\n\t\$1\b} |
| \$1: SNX_SYMBOLIC_CONSTANT__DEFS | Definitions | SNX_SYMB_CONST_LIST (SNX_SYMB_CONST) |

| | | |
|---------------------|----------------------------|-----|
| SNX_SYMB_CONST_LIST | Symbolic Constant Def List | @\n |
|---------------------|----------------------------|-----|

| | | |
|------------------------|-----------------------------|--------------------------|
| SNX_SYMB_CONST | Symbolic Constant Definiton | $\$1\{\$2?\$p4 IS \$2\}$ |
| $\$1$: SNSYMC_DEFINED | Symbolic Constant Defined | NAMED_OBJ_DEF |
| $\$2$: SNSYMC_VALUE | Symbolic Constant Value | literal |

| | | |
|-------------|----------|----------|
| SW_SWITCH_0 | Switch 0 | SWITCH-0 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_1 | Switch 1 | SWITCH-1 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_2 | Switch 2 | SWITCH-2 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_3 | Switch 3 | SWITCH-3 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_4 | Switch 4 | SWITCH-4 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_5 | Switch 5 | SWITCH-5 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_6 | Switch 6 | SWITCH-6 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_7 | Switch 7 | SWITCH-7 |
|-------------|----------|----------|

| | | |
|-------------|----------|----------|
| SW_SWITCH_8 | Switch 8 | SWITCH-8 |
|-------------|----------|----------|

| | | |
|------------------------|------------------|--------------|
| SW_SWITCH | Switch reference | SWITCH $\$1$ |
| $\$1$: SW_SWITCH_NAME | Switch Name | literal |

| | | |
|-----------|---------------|--------|
| SW_UPSI_0 | Switch Upsi 0 | UPSI-0 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_1 | Switch Upsi 1 | UPSI-1 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_2 | Switch Upsi 2 | UPSI-2 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_3 | Switch Upsi 3 | UPSI-3 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_4 | Switch Upsi 4 | UPSI-4 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_5 | Switch Upsi 5 | UPSI-5 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_6 | Switch Upsi 6 | UPSI-6 |
|-----------|---------------|--------|

| | | |
|-----------|---------------|--------|
| SW_UPSI_7 | Switch Upsi 7 | UPSI-7 |
|-----------|---------------|--------|

| | | |
|---|-------------|--|
| <ul style="list-style-type: none"> • io-function | IO function | IOF_CRT IOF_CRT_UNDER IOF_TAB IOF_PRINTER IOF_FORMFEED IOF_COMMAND_LINE IOF_ARG_NUMBER IOF_ARG_VALUE IOF_ENV_NAME IOF_ENV_VALUE IOF_SYSIN IOF_SYSIN IOF_SYSOUT IOF_SYSOUT IOF_SYSOUT IOF_SYSPUNCH IOF_SYSPUNCH IOF_CONSOLE IOF_C01 IOF_C02 IOF_C03 IOF_C04 IOF_C05 IOF_C06 IOF_C07 IOF_C08 IOF_C09 IOF_C010 IOF_C011 IOF_C012 IOF_CSP IOF_S01 IOF_S02 IOF_S03 IOF_S04 IOF_S05 |
|---|-------------|--|

| | | |
|------------------|--------------------------|-------------------|
| IOF_CRT | IOfunc Crt | CRT |
| IOF_CRT_UNDER | IOfunc Crt Under | CRT-UNDER |
| IOF_TAB | IOfunc Tab | TAB |
| IOF_PRINTER | IOfunc Printer | PRINTER |
| IOF_FORMFEED | IOfunc Formfeed | FORMFEED |
| IOF_COMMAND_LINE | IOfunc Command-LINE | COMMAND-LINE |
| IOF_ARG_NUMBER | IOfunc Argument Number | ARGUMENT-NUMBER |
| IOF_ARG_VALUE | IOfunc Argument Value | ARGUMENT-VALUE |
| IOF_ENV_NAME | IOfunc Environment Name | ENVIRONMENT-NAME |
| IOF_ENV_VALUE | IOfunc Environment Value | ENVIRONMENT-VALUE |
| IOF_SYSIN | IOfunc Sysin | SYSIN |
| IOF_SYSOUT | IOfunc Sysout | SYSOUT |
| IOF_SYSPUNCH | IOfunc Syspunch | SYSPUNCH |
| IOF_CONSOLE | IOfunc Console | CONSOLE |
| IOF_C01 | IOfunc C01 | C01 |
| IOF_C02 | IOfunc C02 | C02 |

| | | |
|----------|------------|-----|
| IOF_C03 | IOfunc C03 | C03 |
| IOF_C04 | IOfunc C04 | C04 |
| IOF_C05 | IOfunc C05 | C05 |
| IOF_C06 | IOfunc C06 | C06 |
| IOF_C07 | IOfunc C07 | C07 |
| IOF_C08 | IOfunc C08 | C08 |
| IOF_C09 | IOfunc C09 | C09 |
| IOF_C010 | IOfunc C10 | C10 |
| IOF_C011 | IOfunc C11 | C11 |
| IOF_C012 | IOfunc C12 | C12 |
| IOF_CSP | IOfunc Csp | CSP |
| IOF_S01 | IOfunc S01 | S01 |
| IOF_S02 | IOfunc S02 | S02 |
| IOF_S03 | IOfunc S03 | S03 |
| IOF_S04 | IOfunc S04 | S04 |
| IOF_S05 | IOfunc S05 | S05 |

10.4.2 INPUT-OUTPUT Section

| | | |
|----------------------|----------------------|--|
| SECTION_INPUT_OUTPUT | Input-Output Section | { \$1? \n \$1 } { \$2? \$2 } { \$3? \n \$3 } |
| \$1: IO_HEADER | Header | SECTION_INPUT_OUTPUT_HEADER NULL |
| \$2: IO_FILE_CONTROL | File Control | PARA_FILE_CONTROL NULL |
| \$3: IO_IO_CONTROL | I-O Control | PARA_I_O_CONTROL NULL |

| | | |
|-----------------------------|----------------------|-----------------------|
| SECTION_INPUT_OUTPUT_HEADER | Input-Output Section | INPUT-OUTPUT SECTION. |
|-----------------------------|----------------------|-----------------------|

FILE-CONTROL paragraph

| | | |
|-------------------|----------------------|--|
| PARA_FILE_CONTROL | File Control Para | { \$1? \n \$1 } { \$2? \n \t \$2 \b } |
| \$1: FC_HEADER | Header | PARA_FILE_CONTROL_HEADER NULL |
| \$2: FC_ENTRIES | File Control Entries | DECL_FC_ENTRY_LIST (DECL_FC_ENTRY) NULL |

| | | |
|--------------------------|--------------------------|---------------|
| PARA_FILE_CONTROL_HEADER | File Control Para Header | FILE-CONTROL. |
|--------------------------|--------------------------|---------------|

| | | |
|--------------------|-------------------------|---------|
| DECL_FC_ENTRY_LIST | File Control Entry List | \r @ \n |
|--------------------|-------------------------|---------|

| | | |
|-------------------------|---------------------------|--|
| DECL_FC_ENTRY | File Control Entry | \r SELECT { \$1? \$1 } { \$2= \$2 } { \$3? \n \t \$3 \b } |
| \$1: FC_OPTIONAL | Optional / Not Optional | FCX_OPTIONAL FCX_NOT_OPTIONAL NULL |
| \$2: FC_FILE_NAME | File Name | NAMED_OBJ_USE |
| \$3: FC_ASSIGN | Assign To | FCX_ASSIGN NULL |
| \$4: FC_ORGANIZATION | Organization | FCX_ORGANIZATION NULL |
| \$5: FC_ACCESS_MODE | Access Mode | FCX_ACCESS_MODE NULL |
| \$6: FC_RECORD_KEY | Record Key | FCX_RECORD_KEY NULL |
| \$7: FC_ALT_RECORD_KEYS | Alternate Record Key | FCX_ALTERNATE_RECORD_KEY_LIST (FCX_ALTERNATE_RECORD_KEY FCX_WANG_ALT_REC_KEY) NULL |
| \$8: FC_FILE_STATUS | File Status | FCX_FILE_STATUS NULL |
| \$9: FC_LOCK_MODE | Lock Mode | FCX_LOCK_MODE_IS NULL |
| \$10: FC_PASSWORD | Password | FCX_PASSWORD NULL |
| \$11: FC_PADDING_CHAR | Padding Character | FCX_PADDING_CHAR NULL |
| \$12: FC_REC_DELIMITER | Record Delimiter | FCX_REC_DELIMITER NULL |
| \$13: FC_RESERVE | Reserve N Areas | FCX_RESERVE NULL |
| \$14: FC_CURSOR_POSN | Cursor Position (Wang) | FCX_CURSOR_POSN NULL |
| \$15: FC_BUFFER_SIZE | Buffer Size (Wang) | FCX_BUFFER_SIZE NULL |
| \$16: FC_PFKEY | Pfkey (Wang) | FCX_PFKEY NULL |
| \$17: FC_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|--------------|-----------------|----------|
| FCX_OPTIONAL | Optional phrase | OPTIONAL |
|--------------|-----------------|----------|

| | | |
|------------------|---------------------|--------------|
| FCX_NOT_OPTIONAL | Not Optional phrase | NOT OPTIONAL |
|------------------|---------------------|--------------|

| | | |
|---------------------------|------------------|---------------|
| FCX_ASSIGN | Assign To Clause | ASSIGN TO \$1 |
| \$1: FCX_ASSIGN__FILE_REF | File Reference | FCX_DEVICE |

| | | |
|--------------------------|-----------------------|--|
| FCX_DEVICE | Disk Device | { \$1? \$1 } { \$2? \$2 } { \$3? \$3 } |
| \$1: FCX_DEVICE__EXT_DYN | External/Dynamic | FCX_EXTERNAL FCX_DYNAMIC NULL |
| \$2: FCX_DEVICE__TYPE | Device Type | DEV_DISK_FILE DEV_DISK_FROM_FILE DEV_LINE_ADVANCING DEV_MULTIPLE_REEL DEV_MULTIPLE_UNIT DEV_INPUT DEV_INPUT_OUTPUT DEV_OUTPUT DEV_PRINT DEV_RANDOM DEV_DISK DEV_KEYBOARD DEV_DISPLAY DEV_PRINTER DEV_PRINTER_1 |
| \$3: FCX_DEVICE__REFS | File names/references | EX_LIST (NAMED_OBJ_USE FCX_EXTERNAL_FILE_REF) |

| | | |
|---------------|-----------|--|
| DEV_DISK_FILE | Disk File | |
|---------------|-----------|--|

| | | |
|--------------------|-----------|-----------|
| DEV_DISK_FROM_FILE | Disk From | DISK FROM |
|--------------------|-----------|-----------|

| | | |
|--------------------|---------------------|----------------|
| DEV_LINE_ADVANCING | Line Advancing File | LINE ADVANCING |
|--------------------|---------------------|----------------|

| | | |
|-------------------|--------------------|---------------|
| DEV_MULTIPLE_REEL | Multiple Reel File | MULTIPLE REEL |
|-------------------|--------------------|---------------|

| | | |
|-------------------|--------------------|---------------|
| DEV_MULTIPLE_UNIT | Multiple Unit File | MULTIPLE UNIT |
|-------------------|--------------------|---------------|

| | | |
|-----------|-----------------|-------|
| DEV_INPUT | Input File (RM) | INPUT |
|-----------|-----------------|-------|

| | | |
|---|-------------------------|--|
| DEV_INPUT_OUTPUT | Input-Output File (RM) | INPUT-OUTPUT |
| DEV_OUTPUT | Output File (RM) | OUTPUT |
| DEV_PRINT | Print File (RM) | PRINT |
| DEV_RANDOM | Random File (RM) | RANDOM |
| DEV_DISK | Disk File | DISK |
| DEV_KEYBOARD | Keyboard Device | KEYBOARD |
| DEV_DISPLAY | Display Device | DISPLAY |
| DEV_PRINTER | Printer Device | PRINTER |
| DEV_PRINTER_1 | Printer-1 Device | PRINTER-1 |
| FCX_EXTERNAL | External Phrase | EXTERNAL |
| FCX_DYNAMIC | Dynamic Phrase | DYNAMIC |
| FCX_EXTERNAL_FILE_REF | External File Reference | \$1 |
| \$1: FCX_EXTERNAL_FILE_REF__CHARSTRING | CharString | CHAR.STRING |
| FCX_ORGANIZATION | Organization Clause | ORGANIZATION IS \$1 |
| \$1: FCX_ORGANIZATION__TYPE | Type | FCX_ORG_SEQUENTIAL FCX_ORG_RECORD_SEQUENTIAL FCX_ORG_LINE_SEQUENTIAL FCX_ORG_RELATIVE FCX_ORG_INDEXED FCX_ORG_BINARY_SEQUENTIAL |

| | | |
|------------------------------------|-------------------------------------|--|
| FCX_ORG_SEQUENTIAL | Sequential Organization | SEQUENTIAL |
| FCX_ORG_RECORD_SEQUENTIAL | Record Sequential Organization | RECORD SEQUENTIAL |
| FCX_ORG_LINE_SEQUENTIAL | Line Sequential Organization | LINE SEQUENTIAL |
| FCX_ORG_RELATIVE | Relative Organization | RELATIVE |
| FCX_ORG_INDEXED | Indexed Organization | INDEXED |
| FCX_ORG_BINARY_SEQUENTIAL | Binary Sequential Organization | BINARY SEQUENTIAL |
| FCX_ACCESS_MODE | Access Mode Clause | ACCESS MODE IS \$1{\$2? \$2} |
| \$1: FCX_ACCESS_MODE__MODE | Access Mode | FCX_AM_SEQUENTIAL FCX_AM_RANDOM FCX_AM_DYNAMIC |
| \$2: FCX_ACCESS_MODE__RELKEY | Relative Key | FCX_RELATIVE_KEY NULL |
| FCX_AM_SEQUENTIAL | Sequential Acces Mode | SEQUENTIAL |
| FCX_AM_DYNAMIC | Dynamic Access Mode | DYNAMIC |
| FCX_AM_RANDOM | Random Access Mode | RANDOM |
| FCX_RELATIVE_KEY | Relative Key Phrase | RELATIVE KEY IS \$1 |
| \$1: FCX_RELATIVE_KEY__DATAITEM | Data Item | NAMED_OBJ_USE |
| FCX_RECORD_KEY | Record Key Clause | RECORD KEY IS \$1 |
| \$1: FCX_RECORD_KEY__KEY | Key | FCX_COMP_REC_KEY FCX_SPLIT_REC_KEY |
| FCX_ALTERNATE_RECORD_KEY_LIST | Alternate Record Key Clause List | @\n |

| | | |
|--------------------------|-----------------------------|---|
| FCX_ALTERNATE_RECORD_KEY | Alternate Record Key Clause | ALTERNATE RECORD KEY IS \$1{\$2? \$2}-{\$ |
| \$1: ALTRECKEY_KEY | Record Key | FCX_COMP_REC_KEY FCX_SPLIT_REC_KEY |
| \$2: ALTRECKEY_ORDERBY | Order By (ICobol) | FCX_ALTKEY_ORDERBY_ALPHAUPPER NULL |
| \$3: ALTRECKEY_SUPPRESS | Suppress When (ICobol) | FCX_ALTKEY_SUPPRESS_WHEN NULL |
| \$4: ALTRECKEY_VALUES | Values Are (ICobol) | FCX_ALTKEY_VALUES_ARE NULL |
| \$5: ALTRECKEY_DUPLS | With Duplicates | FCX_ALTKEY_WITH_DUPLICATES NULL |

| | | |
|---------------------------------|------------------------------|--|
| FCX_COMP_REC_KEY | Unnamed Component Record Key | \$1 |
| \$1: FCX_COMP_REC_KEY__COMPS | Component List | FCX_REC_KEY_COMP_LIST (NAMED_OBJ_USE) |

| | | |
|-----------------------|---------------------------|----|
| FCX_REC_KEY_COMP_LIST | Record Key Component List | @& |
|-----------------------|---------------------------|----|

| | | |
|------------------------------------|----------------------------------|---|
| FCX_SPLIT_REC_KEY | Split Named Component Record Key | \$1 ={\$2= \$2}-{\$3? \$3} |
| \$1: FCX_SPLIT_REC_KEY__DEFINED | Name | NAMED_OBJ_DEF |
| \$2: FCX_SPLIT_REC_KEY__COMPS | Component List | SPLIT_KEY_COMP_ALSO_LIST (NAMED_OBJ_USE) SPLIT_KEY_COMP_LIST (NAMED_OBJ_USE) |
| \$3: FCX_SPLIT_REC_KEY__OCCURS | Occurs | SPLIT_KEY_OCCURS NULL |

| | | |
|--------------------------|-------------------------|---------|
| SPLIT_KEY_COMP_ALSO_LIST | ALSO Key Component List | @%ALSO& |
|--------------------------|-------------------------|---------|

| | | |
|---------------------|--------------------|----|
| SPLIT_KEY_COMP_LIST | Key Component List | @& |
|---------------------|--------------------|----|

| | | |
|---------------------------------|------------------|------------------|
| SPLIT_KEY_OCCURS | Split Key Occurs | OCCURS \$1 TIMES |
| \$1: SPLIT_KEY_OCCURS__TIMES | Times | literal |

| | | |
|-------------------------------|-----------------------------|----------------------|
| FCX_ALTKEY_ORDERBY_ALPHAUPPER | Order By Alpha-Upper Phrase | ORDER BY ALPHA-UPPER |
|-------------------------------|-----------------------------|----------------------|

| | | |
|---|----------------------|-------------------|
| FCX_ALTKEY_SUPPRESS_WHEN | Suppress When Phrase | SUPPRESS WHEN \$1 |
| \$1: FCX_ALTKEY_SUPPRESS_WHEN__LITERAL | Literal | literal |

| | | |
|--|----------------------|---------------------------------|
| FCX_ALTKEY_VALUES_ARE | Values Are Phrase | VALUES ARE \$1 |
| \$1: FCX_ALTKEY_VALUES_ARE__ASCDESC | Ascending/Descending | EX_ASCENDING EX_DESCENDING |

| | | |
|----------------------------|------------------------|-----------------|
| FCX_ALTKEY_WITH_DUPLICATES | With Duplicates Phrase | WITH DUPLICATES |
|----------------------------|------------------------|-----------------|

| | | |
|-----------------------------|------------------------------|--------------------------------|
| FCX_FILE_STATUS | File Status Clause | FILE STATUS IS \$1{\$2? \$2} |
| \$1: FCX_FILE_STATUS__NAME1 | File Status Data Item | NAMED_OBJ_USE |
| \$2: FCX_FILE_STATUS__NAME2 | Extra File Status Data Items | EX_LIST (NAMED_OBJ_USE) NULL |

| | | |
|---------------------------------|------------------|--------------------|
| FCX_LOCK_MODE_IS | Lock Mode Clause | LOCK MODE IS \$1 |
| \$1: FCX_LOCK_MODE_IS__DESCR | Descriptor | FCX_LOCK_MODE_WITH |

| | | |
|-----------------------------------|----------------------|--|
| FCX_LOCK_MODE_WITH | Lock Mode Descriptor | \$1{\$2? WITH \$2} |
| \$1: FCX_LOCK_MODE_WITH__PART1 | Part 1 | FCX_MANUAL FCX_AUTOMATIC FCX_EXCLUSIVE |
| \$2: FCX_LOCK_MODE_WITH__PART2 | Part 2 | FCX_LOCK_ON_RECORD FCX_LOCK_ON_MULT_RECS FCX_ROLLBACK NULL |

| | | |
|------------|------------------|--------|
| FCX_MANUAL | Manual Lock Mode | MANUAL |
|------------|------------------|--------|

| | | |
|---------------|---------------------|-----------|
| FCX_AUTOMATIC | Automatic Lock Mode | AUTOMATIC |
|---------------|---------------------|-----------|

| | | |
|---------------|---------------------|-----------|
| FCX_EXCLUSIVE | Exclusive Lock Mode | EXCLUSIVE |
|---------------|---------------------|-----------|

| | | |
|--------------------|--------------------------|----------------|
| FCX_LOCK_ON_RECORD | Lock On Record Qualifier | LOCK ON RECORD |
|--------------------|--------------------------|----------------|

| | | |
|-----------------------|------------------------------------|--------------------------|
| FCX_LOCK_ON_MULT_RECS | Lock On Multiple Records Qualifier | LOCK ON MULTIPLE RECORDS |
|-----------------------|------------------------------------|--------------------------|

| | | |
|--------------|--------------------|----------|
| FCX_ROLLBACK | Rollback Qualifier | ROLLBACK |
|--------------|--------------------|----------|

| | | |
|--------------------------------|-----------------|-----------------|
| FCX_PASSWORD | Password Clause | PASSWORD IS \$1 |
| \$1: FCX_PASSWORD__DATAITEM | Data Item | NAMED_OBJ_USE |

| | | |
|---------------------------------|--------------------------|--------------------------|
| FCX_PADDING_CHAR | Padding Character Clause | PADDING CHARACTER IS \$1 |
| \$1: FCX_PADDING_CHAR__VALUE | Value | ident-liter |

| | | |
|----------------------------------|-------------------------|-------------------------------|
| FCX_REC_DELIMITER | Record Delimiter Clause | RECORD DELIMITER IS \$1 |
| \$1: FCX_REC_DELIMITER__VALUE | Value | FCX_STANDARD_1 STR_CONST |

| | | |
|----------------|-----------------------------|------------|
| FCX_STANDARD_1 | Standard-1 Record Delimiter | STANDARD-1 |
|----------------|-----------------------------|------------|

| | | |
|-------------------------|----------------------|--------------------------|
| FCX_RESERVE | Reserve Areas Clause | RESERVE \$1[AREAS] |
| \$1: FCX_RESERVE__VALUE | Value | EX_NO INT_NUM_CONST |

| | | |
|-------|----------|----|
| EX_NO | No Areas | NO |
|-------|----------|----|

I-O-CONTROL paragraph

| | | |
|-----------------------------------|-----------------------|--|
| PARA_I_O_CONTROL | I-O-Control Paragraph | I-O-CONTROL.{ \$1?\n\t\$1.\b} |
| \$1: PARA_I_O_CONTROL__CLAUSES | Clause List | STMT_LIST (IOCX_RERUN IOCX_SAME_AREA IOCX_MULTIPLE_FILE IOCX_APPLY) |

| | | |
|------------------------|---------------|---|
| IOCX_RERUN | Rerun Clause | RERUN{ \$1? \$1}{ \$2= EVERY \$2} |
| \$1: IOC_RERUN_CLAUSES | Clause List | IOCX_RERUN_ON NULL |
| \$2: IOC_RERUN_ENDCOND | End Condition | IOCX_END_OF IOCX_RECORDS_OF IOCX_CLOCK_UNITS NULL |

| | | |
|--------------------------|--------------------|---------------|
| IOCX_RERUN_ON | Rerun On File Name | ON \$1 |
| \$1: IOCX_RERUN_ON__FILE | File Name | NAMED_OBJ_USE |

| | | |
|---------------------|----------------------------|--------------------------|
| IOCX_END_OF | End Of Reel/Unit Condition | END OF \$1{\$2= OF \$2} |
| \$1: IOC_ENDOF_WHAT | Reel/Unit | IOCX_REEL IOCX_UNIT |
| \$2: IOC_ENDOF_FILE | File Name | NAMED_OBJ_USE |

| | | |
|--------------------------|-----------------------------|--------------------------|
| IOCX_RECORDS_OF | Number Of Records Condition | \$1{\$2= RECORDS OF \$2} |
| \$1: IOC_RECISOFF_NUMBER | Number Of Recs | INT_NUM_CONST |
| \$2: IOC_RECISOFF_FILE | File Name | NAMED_OBJ_USE |

| | | |
|----------------------------------|----------------------------|-------------------|
| IOCX_CLOCK_UNITS | Number Of Clock Units Cond | \$1[CLOCK-UNITS] |
| \$1: IOCX_CLOCK_UNITS__NUMBER | Number | INT_NUM_CONST |

| | | |
|-------------------|------------------|---|
| IOCX_SAME_AREA | Same Area Clause | SAME{\$1? \$1}{\$2= AREA FOR \$2} |
| \$1: IOC_SA_TYPE | Area Type | IOCX_RECORD IOCX_SORT IOCX_SORT_MERGE |
| \$2: IOC_SA_FILES | File Names | EX_LIST (NAMED_OBJ_USE) |

| | | |
|-------------|------------------|--------|
| IOCX_RECORD | Record Area Type | RECORD |
|-------------|------------------|--------|

| | | |
|-----------|----------------|------|
| IOCX_SORT | Sort Area Type | SORT |
|-----------|----------------|------|

| | | |
|-----------------|----------------------|------------|
| IOCX_SORT_MERGE | Sort-Merge Area Type | SORT-MERGE |
|-----------------|----------------------|------------|

| | | |
|---------------------------------------|---------------------------|---------------------------------|
| IOCX_MULTIPLE_FILE | Multiple File Tape Clause | MULTIPLE FILE TAPE CONTAINS \$1 |
| \$1: IOCX_MULTIPLE_FILE__FILEPOSNS | File Positions | EX_LIST (IOCX_FILE_POSN) |

| | | |
|---------------------|---------------|----------------------|
| IOCX_FILE_POSN | File Position | \$1{\$2? \$2} |
| \$1: IOC_FPOSN_FILE | File Name | NAMED_OBJ_USE |
| \$2: IOC_FPOSN_POSN | Position | IOCX_POSITION NULL |

| | | |
|----------------------------|----------|---------------|
| IOCX_POSITION | Position | POSITION \$1 |
| \$1: IOCX_POSITION__NUMBER | Number | INT_NUM_CONST |

| | | |
|----------------------|---------------|---|
| IOCX_APPLY | Apply Clause | APPLY \$1{\$2= ON \$2} |
| \$1: IOC_APPLY_TYPE | Apply Type | IOCX_WRITE_ONLY IOCX_CORE_INDEX IOCX_RECORD_OVERFLOW IOCX_REORG_CRITERIA |
| \$2: IOC_APPLY_FILES | On File Names | EX_LIST (NAMED_OBJ_USE) |

| | | |
|-----------------|-----------------------|------------|
| IOCX_WRITE_ONLY | Write-Only Apply Type | WRITE-ONLY |
|-----------------|-----------------------|------------|

| | | |
|-----------------|-----------------------|------------|
| IOCX_CORE_INDEX | Core-Index Apply Type | CORE-INDEX |
|-----------------|-----------------------|------------|

| | | |
|----------------------|----------------------------|-----------------|
| IOCX_RECORD_OVERFLOW | Record-Overflow Apply Type | RECORD-OVERFLOW |
|----------------------|----------------------------|-----------------|

| | | |
|---------------------|---------------------------|----------------|
| IOCX_REORG_CRITERIA | Reorg-Criteria Apply Type | REORG-CRITERIA |
|---------------------|---------------------------|----------------|

10.5 DATA Division

| | | |
|---------------------------|-------------------------------|---|
| DIVISION_DATA | Data Division | { \$1? \n\$1 } { \$2? \n\$2 } { \$3? \n\$3 } { \$4? \$4 } { \$5? \n\$5 } { \$6? \n\$6 } { \$7? \n\$7 } { \$8? \n\$8 } { \$9? \n\$9 } { \$10? \n\$10 } |
| \$1: DS_HEADER | Data Division Header | DIVISION_DATA_HEADER NULL |
| \$2: DS_SYM_CONST_REPLACE | Replace Stmt for SymConstants | STMT_REPLACE NULL |
| \$3: DS_SCHEMA | Schema Section (DML) | SECTION_SCHEMA NULL |
| \$4: DS_FILE | File Section | SECTION_FILE NULL |
| \$5: DS_WORKING_STORAGE | Working-Storage Section | SECTION_WORKING_STORAGE NULL |
| \$6: DS_LOCAL_STORAGE | Local-Storage Section (MF) | SECTION_LOCAL_STORAGE NULL |
| \$7: DS_LINKAGE | Linkage Section | SECTION_LINKAGE NULL |
| \$8: DS_COMMUNICATION | Communication Section | SECTION_COMMUNICATION NULL |
| \$9: DS_REPORT | Report Section | SECTION_REPORT NULL |
| \$10: DS_SCREEN | Screen Section | SECTION_SCREEN NULL |

| | | |
|----------------------|----------------------|----------------|
| DIVISION_DATA_HEADER | Data Division Header | DATA DIVISION. |
|----------------------|----------------------|----------------|

10.5.1 FILE Section

| | | |
|---------------------------|---------------------|-------------------------------|
| SECTION_FILE | File Section | { \$1? \n\$1 } { \$2? \n\$2 } |
| \$1: SECTION_FILE_HEADER | File Section Header | SECTION_FILE_HEADER NULL |
| \$2: SECTION_FILE_ENTRIES | FD/SD entries | DECL_FD_LIST (DECL_FD) NULL |

| | | |
|---------------------|---------------------|---------------|
| SECTION_FILE_HEADER | File Section Header | FILE SECTION. |
|---------------------|---------------------|---------------|

| | | |
|--------------|---------------|--------|
| DECL_FD_LIST | FD Entry List | \r@ \n |
|--------------|---------------|--------|

| | | |
|----------------|-------------------------------|-------------------------------------|
| DECL_FD | File Description Entry Holder | \r\$1 { \$2? \n\$2 } |
| \$1: FDH_ENTRY | FD Entry | DECL_FD_ENTRY copy-expr NULL |
| \$2: FDH_ARGS | Record Description Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|-------------------------|---------------------------|--|
| DECL_FD_ENTRY | File Description Entry | \r\$1 ~x{\$2=\$2}{\$3?\n\$x\$3}{\$4?\n\$x\$4} |
| \$1: FD_LEVEL | Level Indicator | FDX_LEVEL SDX_LEVEL |
| \$2: FD_FILE_NAME | File Name Defined | NAMED_OBJ_DEF |
| \$3: FD_RECORD_SIZE | Record Size | FDX_RECORD_CONTAINS FDX_RECORD_VARYING NULL |
| \$4: FD_REC_MODE | Recording Mode | FDX_REC_MODE NULL |
| \$5: FD_LINAGE | Linage | FDX_LINAGE NULL |
| \$6: FD_CODE_SET | Code-Set | FDX_CODE_SET NULL |
| \$7: FD_REPORT | Reports | FDX_REPORT NULL |
| \$8: FD_EXTERNAL | External | DX_EXTERNAL NULL |
| \$9: FD_GLOBAL | Global | DX_GLOBAL NULL |
| \$10: FD_BLOCK_SIZE | Block Contains | FDX_BLOCK_CONTAINS NULL |
| \$11: FD_DATA_RECORDS | Data Records | FDX_DATA_RECORDS NULL |
| \$12: FD_LABEL_RECORDS | Label Records | FDX_LABEL_RECORDS NULL |
| \$13: FD_VALUE_OF | Value Of | FDX_VALUE_OF_LIST (FDX_VALUE_OF) NULL |
| \$14: FD_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|-----------|----------|----|
| FDX_LEVEL | FD level | FD |
|-----------|----------|----|

| | | |
|-----------|----------|----|
| SDX_LEVEL | SD level | SD |
|-----------|----------|----|

| | | |
|--|------------------------|--|
| FDX_RECORD_CONTAINS | Record Contains Clause | RECORD CONTAINS \$1{\$2? \$2}[CHARACTER |
| \$1: FDX_RECORD_CONTAINS__LEN_RANGE | Record Length Range | EX_INT_CONST_RANGE |
| \$2: FDX_RECORD_CONTAINS__COMPR | Compressed (Wang) | FDX_COMPRESSED NULL |

| | | |
|---------------------|----------------------|----------------------|
| EX_INT_CONST_RANGE | Integer Range Phrase | \$1{\$2? TO \$2} |
| \$1: INT_RANGE_FROM | From | INT_NUM_CONST |
| \$2: INT_RANGE_TO | To | INT_NUM_CONST NULL |

| | | |
|------------------------------------|--------------------------|---|
| FDX_RECORD_VARYING | Record Is Varying Clause | RECORD IS VARYING IN SIZE{\$1? \$1}{\$2 |
| \$1: FDX_RECORD_VARYING__FROM | From What | FDX_FROM NULL |
| \$2: FDX_RECORD_VARYING__TO | To What | FDX_TO NULL |
| \$3: FDX_RECORD_VARYING__COMPR | Compressed (Wang) | FDX_COMPRESSED NULL |
| \$4: FDX_RECORD_VARYING__DEPEND | Depending On | EX_DEPENDING_ON NULL |

| | | |
|----------------------|-----------|---------------|
| FDX_FROM | FD From | FROM \$1 |
| \$1: FDX_FROM__VALUE | From What | INT_NUM_CONST |

| | | |
|--------------------|---------|---------------|
| FDX_TO | FD To | TO \$1 |
| \$1: FDX_TO__VALUE | To What | INT_NUM_CONST |

| | | |
|-----------------------------------|-----------------------|--|
| FDX_BLOCK_CONTAINS | Block Contains Clause | BLOCK CONTAINS \$1{\$2? \$2} |
| \$1: FDX_BLOCK_CONTAINS__RANGE | Length Range | EX_INT_CONST_RANGE |
| \$2: FDX_BLOCK_CONTAINS__UNITS | Length Units | FDX_RECORDS FDX_CHARACTERS NULL |

| | | |
|-------------|--------------|---------|
| FDX_RECORDS | Records Unit | RECORDS |
|-------------|--------------|---------|

| | | |
|----------------|-----------------|------------|
| FDX_CHARACTERS | Characters Unit | CHARACTERS |
|----------------|-----------------|------------|

| | | |
|-----------------------------------|---------------------|------------------|
| EX_DEPENDING_ON | Depending On Phrase | DEPENDING ON \$1 |
| \$1: EX_DEPENDING_ON__DATAITEM | Data Item | NAMED_OBJ_USE |

| | | |
|-------------------------|-----------------------|---|
| FDX_REC_MODE | Recording Mode Clause | RECORDING MODE IS \$1 |
| \$1: FDX_REC_MODE__MODE | Recording Mode | FDX_MODE_FIXED FDX_MODE_VARIABLE FDX_MODE_U FDX_MODE_S |

| | | |
|----------------|----------------|---|
| FDX_MODE_FIXED | Rec Mode Fixed | F |
|----------------|----------------|---|

| | | |
|-------------------|-------------------|---|
| FDX_MODE_VARIABLE | Rec Mode Variable | V |
|-------------------|-------------------|---|

| | | |
|------------|------------|---|
| FDX_MODE_S | Rec Mode S | S |
|------------|------------|---|

| | | |
|------------|------------|---|
| FDX_MODE_U | Rec Mode U | U |
|------------|------------|---|

| | | |
|----------------------------|------------------------|---|
| FDX_LINAGE | Linage Clause | LINAGE IS \$1 LINES{\$2? \$2}{\$3? \$3}{\$4? \$4} |
| \$1: FDX_LINAGE__PAGE_SIZE | Size Of Page | ident-liter |
| \$2: FDX_LINAGE__FOOTING | Footing At | FDX_LIN_FOOTING NULL |
| \$3: FDX_LINAGE__TOP | Lines At Top | FDX_LIN_TOP NULL |
| \$4: FDX_LINAGE__BOTTOM | Lines At Bottom | FDX_LIN_BOTTOM NULL |
| \$5: FDX_LINAGE__COUNTER | Linage-Counter Defined | NAMED_OBJ_DEF |

| | | |
|-----------------------------|-------------------|---------------------|
| FDX_LIN_FOOTING | Footing At Phrase | WITH FOOTING AT \$1 |
| \$1: FDX_LIN_FOOTING__LINES | Lines | ident-liter |

| | | |
|-------------------------|---------------------|------------------|
| FDX_LIN_TOP | Lines At Top Phrase | LINES AT TOP \$1 |
| \$1: FDX_LIN_TOP__LINES | Lines | ident-liter |

| | | |
|----------------------------|------------------------|---------------------|
| FDX_LIN_BOTTOM | Lines At Bottom Phrase | LINES AT BOTTOM \$1 |
| \$1: FDX_LIN_BOTTOM__LINES | Lines | ident-liter |

| | | |
|-----------------------------|-------------------|---------------------------|
| FDX_CODE_SET | Code Set Clause | CODE-SET IS \$1{\$2? \$2} |
| \$1: FDX_CODE_SET__ALPHABET | Alphabet | NAMED_OBJ_USE |
| \$2: FDX_CODE_SET__FOR | For Idents Phrase | FDX_CODE_SET_FOR NULL |

| | | |
|----------------------------------|-------------------|-------------------------|
| FDX_CODE_SET_FOR | For Idents Phrase | FOR \$1 |
| \$1: FDX_CODE_SET_FOR__IDENTS | Identifiers | IDENT_LIST (identifier) |

| | | |
|--------------------------|---------------|-------------------------|
| FDX_REPORT | Report Clause | REPORTS ARE \$1 |
| \$1: FDX_REPORT__REPORTS | Reports | EX_LIST (NAMED_OBJ_USE) |

| | | |
|-------------------------------------|---------------------|-------------------------|
| FDX_DATA_RECORDS | Data Records Clause | DATA RECORDS ARE \$1 |
| \$1: FDX_DATA_RECORDS__DATAITEMS | Data Items | EX_LIST (NAMED_OBJ_USE) |

| | | |
|--------------------------------------|----------------------|--|
| FDX_LABEL_RECORDS | Label Records Clause | LABEL RECORDS ARE \$1 |
| \$1: FDX_LABEL_RECORDS__DATAITEMS | Data Items | FDX_STANDARD FDX_OMITTED EX_LIST (NAMED_OBJ_USE) |

| | | |
|--------------|------------------------|----------|
| FDX_STANDARD | Standard Label Records | STANDARD |
|--------------|------------------------|----------|

| | | |
|-------------|-----------------------|---------|
| FDX_OMITTED | Omitted Label Records | OMITTED |
|-------------|-----------------------|---------|

| | | |
|-------------------|----------------------|-----|
| FDX_VALUE_OF_LIST | Value Of Clause List | @\n |
|-------------------|----------------------|-----|

| | | |
|--------------------------|-----------------|---|
| FDX_VALUE_OF | Value Of Clause | VALUE OF \$1 |
| \$1: FDX_VALUE_OF__ITEMS | Items | FDX_VALUE_OF_ITEM_LIST (FDX_VALUE_OF_ITEM) |

| | | |
|------------------------|--------------------|-----|
| FDX_VALUE_OF_ITEM_LIST | Value Of Item List | @\n |
|------------------------|--------------------|-----|

| | | |
|----------------------------------|------------------|--|
| FDX_VALUE_OF_ITEM | Value Of Item | \$1{\$2= IS \$2} |
| \$1: FDX_VALUE_OF_ITEM__NAME | Valued Data Item | NAMED_OBJ_USE FDX_VO_FILE_ID FDX_VO_FILENAME FDX_VO_LIBRARY FDX_VO_VOLUME FDX_VO_SPACE FDX_VO_POSITION FDX_VO_INDEX_AREA FDX_VO_DATA_AREA FDX_VO_PRINT_CLASS FDX_VO_FILE_CLASS FDX_VO_EXTENT_SIZE FDX_VO_RECOVERY_BLOCKS FDX_VO_RECOVERY_STATUS FDX_VO_DATABASE_NAME |
| \$2: FDX_VALUE_OF_ITEM__VALUE | Value | ident-liter |

| | | |
|----------------|---------|---------|
| FDX_VO_FILE_ID | File-Id | FILE-ID |
|----------------|---------|---------|

10.5.2 Data Storage Sections

| | | |
|--|--------------------------|---|
| SECTION_WORKING_STORAGE | Working-Storage Section | WORKING-STORAGE SECTION.{ \$1? \n \$1 } |
| \$1: SECTION_WORKING_STORAGE__ENTRIES | Data Description Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|--|--------------------------|---------------------------------------|
| SECTION_LOCAL_STORAGE | Local-Storage Section | LOCAL-STORAGE SECTION.{ \$1? \n \$1 } |
| \$1: SECTION_LOCAL_STORAGE__ENTRIES | Data Description Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|----------------------------------|--------------------------|---------------------------------|
| SECTION_LINKAGE | Linkage Section | LINKAGE SECTION.{ \$1? \n \$1 } |
| \$1: SECTION_LINKAGE__ENTRIES | Data Description Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|------------|------------------|------|
| DECL_NO_OP | No-Op (data div) | \377 |
|------------|------------------|------|

| | | |
|--------------|---------------------------|-------|
| DECL_DD_LIST | Data Item Decl Entry List | \r@\n |
|--------------|---------------------------|-------|

| | | |
|----------------|-----------------------------|--|
| DECL_DD | Data Item Decl Entry Holder | \r\$1{\$2?\n\t\$2\b} |
| \$1: DDH_ENTRY | DD Entry | DECL_DD_ENTRY DECL_CONST_ENTRY DECL_COND_ENTRY STMT_EXEC_IN_DCL copy-expr NULL |
| \$2: DDH_ARGS | Subordinate DD Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|---------------------------|------------------------------------|--|
| DECL_DD_HIDDEN | Hidden Data Item Decl Entry Holder | |
| \$1: DECL_DD_HIDDEN_ENTRY | DD Entry | DECL_DD_ENTRY DECL_CONST_ENTRY DECL_COND_ENTRY STMT_EXEC_IN_DCL copy-expr NULL |
| \$2: DECL_DD_HIDDEN_ARGS | Subordinate DD Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|-------------------------|---------------------------|--|
| DECL_DD_ENTRY | Data Item Decl Entry | \r\$1{\$2? \$2}{\$3? \$3}{\$4?\$p4 \$4}{\$5 |
| \$1: DD_LEVEL | Level Number | INT_NUM_CONST |
| \$2: DD_COPY_BEF_NAME | Copy before name | copy-expr NULL |
| \$3: DD_DATA_NAME | Data Item Defined | NAMED_OBJ_DEF EX_FILLER NULL |
| \$4: DD_REDEFINES | Redefines Clause | DX_REDEFINES NULL |
| \$5: DD_PICTURE | Picture | DX_PICTURE NULL |
| \$6: DD_USAGE | Usage | DX_USAGE NULL |
| \$7: DD_BWZ | Blank When Zero | DX_BWZ NULL |
| \$8: DD_JUSTIFIED | Justified | DX_JUSTIFIED NULL |
| \$9: DD_SIGN | Sign | DX_SIGN NULL |
| \$10: DD_SYNCHRONIZED | Synchronized | DX_SYNCHRONIZED NULL |
| \$11: DD_EXTERNAL | External | DX_EXTERNAL NULL |
| \$12: DD_GLOBAL | Global | DX_GLOBAL NULL |
| \$13: DD_RENAMES | Renames | DX_RENAMES NULL |
| \$14: DD_OCCURS | Occurs | DX_OCCURS NULL |
| \$15: DD_VALUE | Value | DX_VALUE NULL |
| \$16: DD_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$17: DD_CATEGORY | Category | DCAT_GROUP DCAT_INDEX DCAT_POINTER DCAT_PROC_POINTER PIC_ALPHA PIC_NUM PIC_ALPHA_NUM PIC_ALPHA_NUM_EDITED PIC_NUM_EDITED PIC_INT_FLOAT PIC_EXT_FLOAT PIC_BIT PIC_NATIONAL PIC_NATIONAL_EDITED |
| \$18: DD_LENGTH | Length | INT_NUM_CONST NULL |
| \$19: DD_OFFSET_LOCAL | Offset From Upper Level | INT_NUM_CONST NULL |
| \$20: DD_OFFSET_01 | Offset From 01 Level | INT_NUM_CONST NULL |
| \$21: DD_OFFSET_GLOBAL | Offset Global | INT_NUM_CONST NULL |

| | | |
|-----------|-------------|--------|
| EX_FILLER | Filler Name | FILLER |
|-----------|-------------|--------|

| | | |
|------------------------|----------------|-------------|
| DX_PICTURE | Picture Clause | PIC \$1 |
| \$1: DX_PICTURE_STRING | Char String | CHAR_STRING |

| | | |
|------------|-------|---|
| DCAT_GROUP | Group | G |
|------------|-------|---|

| | | |
|-----------------------|------------------|----|
| DCAT_BEGIN_ELEMENTARY | Begin Elementary | E< |
|-----------------------|------------------|----|

| | | |
|----------------------|---------------------|----|
| DCAT_INDEX | Index | I |
| DCAT_POINTER | Pointer | P |
| DCAT_PROC_POINTER | Procedure Pointer | PP |
| PIC_ALPHA | Alphabetic | A |
| PIC_NUM | Numeric | N |
| PIC_ALPHA_NUM | AlphaNumeric | AN |
| PIC_ALPHA_NUM_EDITED | AlphaNumeric Edited | AE |
| PIC_NUM_EDITED | Numeric Edited | NE |
| PIC_INT_FLOAT | Internal FLoat | IF |
| PIC_EXT_FLOAT | External FLoat | EF |
| PIC_BIT | Bit | B |
| PIC_NATIONAL | National | J |
| PIC_NATIONAL_EDITED | National Edited | JE |
| DCAT__END_ELEMENTARY | End Elementary | E> |

| | | |
|---------------------|------------------------------|--|
| DX_USAGE | Usage Clause | \$1 |
| \$1: DX_USAGE__TYPE | Usage | DU_DISPLAY DU_DISPLAY_1 DU_DISPLAY_WS DU_COMP DU_COMP_RM DU_COMP_0 DU_COMP_RM DU_BINARY DU_COMP_1 DU_COMP_1_RM DU_COMP_2 DU_COMP_3 DU_PACKED_DECIMAL DU_COMP_4 DU_COMP_5 DU_COMP_6_RM DU_COMP_X DU_BIT DU_INDEX DU_POINTER DU_PROC_POINTER |
| DU_NONE | None Usage (Display assumed) | |
| DU_BINARY | Binary Usage | BINARY |
| DU_COMP | Comp Usage | COMP |
| DU_COMP_RM | Comp RM Dspl Usage | COMP |
| DU_COMP_0 | Comp-0 Usage | COMP-0 |
| DU_COMP_1 | Comp-1 Float Usage | COMP-1 |
| DU_COMP_1_RM | Comp-1 RM Bin Usage | COMP-1 |
| DU_COMP_2 | Comp-2 Double Usage | COMP-2 |

| | | |
|-------------------|-------------------------|---------------------------------------|
| DU_COMP_3 | Comp-3 Usage | COMP-3 |
| DU_COMP_4 | Comp-4 Usage | COMP-4 |
| DU_COMP_5 | Comp-5 Usage | COMP-5 |
| DU_COMP_6_RM | Comp-6 RM Usage | COMP-6 |
| DU_COMP_X | Comp-X Usage | COMP-X |
| DU_BIT | Bit (Fsc) Usage | BIT |
| DU_DISPLAY | Display Usage | DISPLAY |
| DU_DISPLAY_1 | Display-1 Usage | DISPLAY-1 |
| DU_INDEX | Index Usage | INDEX |
| DU_PACKED_DECIMAL | Packed-Decimal Usage | PACKED-DECIMAL |
| DU_POINTER | Pointer Usage | POINTER |
| DU_PROC_POINTER | Procedure-Pointer Usage | PROCEDURE-POINTER |
| DX_BWZ | Blank When Zero Clause | BLANK ZERO |
| DX_JUSTIFIED | Justified Right Clause | JUST RIGHT |
| DX_SIGN | Sign Clause | SIGN IS \$1{\$2? \$2} |
| \$1: DX_SIGN__LT | Leading/Trailing | DX_SIGN_LEADING DX_SIGN_TRAILING |
| \$2: DX_SIGN__SEP | Separate | DX_SIGN_SEPARATE NULL |

| | | |
|-----------------|--------------|---------|
| DX_SIGN_LEADING | Leading Sign | LEADING |
|-----------------|--------------|---------|

| | | |
|------------------|---------------|----------|
| DX_SIGN_TRAILING | Trailing Sign | TRAILING |
|------------------|---------------|----------|

| | | |
|------------------|---------------|----------|
| DX_SIGN_SEPARATE | Separate Sign | SEPARATE |
|------------------|---------------|----------|

| | | |
|--------------------------|---------------------|--|
| DX_SYNCHRONIZED | Synchronized Clause | SYNC{\$1? \$1} |
| \$1: DX_SYNCHRONIZED__LR | Left/Right | DX_SYNC_LEFT DX_SYNC_RIGHT NULL |

| | | |
|--------------|-----------|------|
| DX_SYNC_LEFT | Left Sync | LEFT |
|--------------|-----------|------|

| | | |
|---------------|------------|-------|
| DX_SYNC_RIGHT | Right Sync | RIGHT |
|---------------|------------|-------|

| | | |
|-------------|-----------------|----------|
| DX_EXTERNAL | External Clause | EXTERNAL |
|-------------|-----------------|----------|

| | | |
|-----------|---------------|--------|
| DX_GLOBAL | Global Clause | GLOBAL |
|-----------|---------------|--------|

| | | |
|---------------------------------|---------------------|---------------|
| DX_REDEFINES | Redefines Clause | REDEFINES \$1 |
| \$1: DX_REDEFINES__REDEFINED | Redefined Data Item | NAMED_OBJ_USE |

| | | |
|--------------------------|----------------------|-----------------|
| DX_RENAMES | Renames Clause | RENAMES \$1 |
| \$1: DX_RENAMES__RENAMED | Renamed Data Item(s) | EX_RENAMES_THRU |

| | | |
|----------------------------|---------------------|----------------------|
| EX_RENAMES_THRU | Renames Thru Phrase | \$1{\$2?%THRU \$2} |
| \$1: EX_RENAMES_THRU__FROM | From | NAMED_OBJ_USE |
| \$2: EX_RENAMES_THRU__TO | To | NAMED_OBJ_USE NULL |

| | | |
|----------------------------|-----------------------------|--|
| DX_OCCURS | Occurs Clause | OCCURS{\$1? \$1 TIMES}{\$2? \$2}{\$3? \$3} |
| \$1: DX_OCCURS__TIMES | Times | EX_INT_CONST_RANGE NULL |
| \$2: DX_OCCURS__DEPENDING | Depending On | EX_DEPENDING_ON NULL |
| \$3: DX_OCCURS__KEYS | Keys | DX_OCCURS_KEY_LIST (DX_OCCURS_KEY) NULL |
| \$4: DX_OCCURS__INDEXED_BY | Indexed By | DX_INDEXED_BY NULL |
| \$5: DX_OCCURS__ELT_LENGTH | Length of the array element | INT_NUM_CONST NULL |

| | | |
|--------------------|-----------------|---|
| DX_OCCURS_KEY_LIST | Occurs Key List | @ |
|--------------------|-----------------|---|

| | | |
|-----------------------------|----------------------|---------------------------------|
| DX_OCCURS_KEY | Occurs Key | \$1{\$2= KEY IS \$2} |
| \$1: DX_OCCURS_KEY__ASCDESC | Ascending/Descending | EX_ASCENDING EX_DESCENDING |
| \$2: DX_OCCURS_KEY__NAMES | Key Names | EX_LIST (NAMED_OBJ_USE) |

| | | |
|-----------------------------|-------------------|---------------------------------------|
| DX_INDEXED_BY | Indexed By Phrase | INDEXED BY \$1 |
| \$1: DX_INDEXED_BY__INDICES | Indices defined | DX_INDEX_DEFS_LIST (NAMED_OBJ_DEF) |

| | | |
|--------------------|-----------------------|----|
| DX_INDEX_DEFS_LIST | Index Definition List | @& |
|--------------------|-----------------------|----|

| | | |
|--------------|-----------------|-----------|
| EX_ASCENDING | Ascending Order | ASCENDING |
|--------------|-----------------|-----------|

| | | |
|---------------|------------------|------------|
| EX_DESCENDING | Descending Order | DESCENDING |
|---------------|------------------|------------|

| | | |
|------------------------|-----------------|----------------------------|
| DX_VALUE | DD Value Clause | VALUE \$1 |
| \$1: DX_VALUE__LITERAL | Value Literal | literal NAMED_OBJ_USE |

| | | |
|----------------------------|---------------------------|---|
| DECL_COND_ENTRY | Condition Decl Entry | \r\$1{\$2? \$2}{\$3? \$3}{\$4?\$p6 \$4}{\$5 |
| \$1: DDCOND_LEVEL | Level Number | INT_NUM_CONST |
| \$2: DDCOND_COPY_BEf_NAME | Copy before name | copy-expr NULL |
| \$3: DDCOND_NAME | Constant Name Defined | NAMED_OBJ_DEF |
| \$4: DDCOND_VALUE | Constant Value | EX_COND_VALUE |
| \$5: DDCOND_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|-----------------------------------|------------------------|---|
| EX_COND_VALUE | Condition Value Clause | VALUE \$1{\$2? \$2} |
| \$1: EX_COND_VALUE__RANGES | Ranges | EX_VALUE_RANGE_LIST (EX_VALUE_RANGE) |
| \$2: EX_COND_VALUE__WHEN_FALSE | When False (MF) | EX_VALUE_WHEN_FALSE NULL |

| | | |
|---------------------|----------------------------|----|
| EX_VALUE_RANGE_LIST | Condition Value Range List | @& |
|---------------------|----------------------------|----|

| | | |
|---------------------------|-------------|--------------------|
| EX_VALUE_RANGE | Value Range | \$1{\$2?%THRU \$2} |
| \$1: EX_VALUE_RANGE__FROM | From | literal |
| \$2: EX_VALUE_RANGE__TO | To | literal NULL |

| | | |
|------------------------------------|-------------------|----------------|
| EX_VALUE_WHEN_FALSE | When False Phrase | WHEN FALSE \$1 |
| \$1: EX_VALUE_WHEN_FALSE__VALUE | Value | literal |

| | | |
|--------------------------------|---------------------------|---|
| DECL_CONST_ENTRY | Constant Decl Entry | \r\$1{\$2? \$2}{\$3? \$3}{\$4?\$p6 \$4}{\$5 |
| \$1: DDCONST_LEVEL | Level Number | INT_NUM_CONST |
| \$2: DDCONST_COPY_BEFORE_NAME | Copy before name | copy-expr NULL |
| \$3: DDCONST_NAME | Constant Name Defined | NAMED_OBJ_DEF |
| \$4: DDCONST_VALUE | Constant Value | EX_CONST_VALUE |
| \$5: DDCONST_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$6: DDCONST_VALUE_COMPD | Actual Constant Value | literal NULL |

| | | |
|---------------------------------|-----------------------|------------|
| EX_CONST_VALUE | Constant Value Clause | VALUE \$1 |
| \$1: EX_CONST_VALUE__LITERAL | Value Literal | value-expr |

| | | |
|--------------|------------------|---|
| • value-expr | Value Expression | literal NAMED_OBJ_USE EX_VAL_NEXT EX_VAL_START_OF EX_VAL_LENGTH_OF AR_ADD (value-expr) AR_SUB (value-expr) AR_MUL (value-expr) AR_DIV (value-expr) BIT_AND (value-expr) BIT_OR (value-expr) |
|--------------|------------------|---|

| | | |
|-------------|-----------------|------|
| EX_VAL_NEXT | Next Value Term | NEXT |
|-------------|-----------------|------|

| | | |
|-----------------------------|---------------------|--------------|
| EX_VAL_START_OF | Start Of Value Term | START OF \$1 |
| \$1: EX_VAL_START_OF__IDENT | Identifier | identifier |

| | | |
|---------------------------------|----------------------|---------------|
| EX_VAL_LENGTH_OF | Length Of Value Term | LENGTH OF \$1 |
| \$1: EX_VAL_LENGTH_OF__IDENT | Identifier | identifier |

10.5.3 COMMUNICATION Section

| | | |
|-------------------------------------|-----------------------|---------------------------------------|
| SECTION_COMMUNICATION | Communication Section | COMMUNICATION SECTION.{ \$1? \n \$1 } |
| \$1: SECTION_COMMUNICATION__ENTRIES | CD entries | DECL_CD_ENTRY_LIST (DECL_CD_ENTRY) |

| | | |
|--------------------|---------------|--------|
| DECL_CD_ENTRY_LIST | CD Entry List | \r@ \n |
|--------------------|---------------|--------|

| | | |
|---------------------|------------------------|---|
| DECL_CD_ENTRY | CD Entry | CD \$1 FOR{ \$2? \$2 }{ \$3= \$3 }{ \$4? \n }{ \$4? \n } |
| \$1: CDE_DEFINED | Entry Name Defined | NAMED_OBJ_USE |
| \$2: CDE_INITIAL | Initial | EX_CD_INITIAL NULL |
| \$3: CDE_IO | IO type | EX_CD_INPUT EX_CD_OUTPUT EX_CD_I_O |
| \$4: CDE_PHRASES | Data Item Definitions | EX_CD_LIST (EX_CD_QUEUE_IS EX_CD_MSG_DATE_IS EX_CD_MSG_TIME_IS EX_CD_TEXT_LENGTH_IS EX_CD_END_KEY_IS EX_CD_STATUS_KEY_IS EX_CD_ERROR_KEY_IS EX_CD_MSG_COUNT_IS EX_CD_DEST_COUNT_IS EX_CD_DEST_TABLE_OCCURS) EX_LIST (NAMED_OBJ_USE EX_FILLER) NULL |
| \$5: CDE_DD_ENTRIES | Subordinate DD Entries | DECL_DD_LIST (DECL_DD) NULL |

| | | |
|---------------|---------|---------|
| EX_CD_INITIAL | Initial | INITIAL |
|---------------|---------|---------|

| | | |
|-------------|-------|-------|
| EX_CD_INPUT | Input | INPUT |
|-------------|-------|-------|

| | | |
|--------------|--------|--------|
| EX_CD_OUTPUT | Output | OUTPUT |
|--------------|--------|--------|

| | | |
|-----------|-----|-----|
| EX_CD_I_O | I-O | I-O |
|-----------|-----|-----|

| | | |
|------------|----------------|------|
| EX_CD_LIST | CD clause list | @ \n |
|------------|----------------|------|

| | | |
|----------------|-------------------|---|
| EX_CD_QUEUE_IS | Symbolic Clause | SYMBOLIC \$1{\$2= IS \$2} |
| \$1: CDSQ_TYPE | Type | EX_CD_SYMB_QUEUE EX_CD_SYMB_SUB_QUEUE_1 EX_CD_SYMB_SUB_QUEUE_2 EX_CD_SYMB_SUB_QUEUE_3 EX_CD_SYMB_SOURCE EX_CD_SYMB_DEST EX_CD_SYMB_TERMINAL |
| \$2: CDSQ_DEFD | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|------------------|-------|-------|
| EX_CD_SYMB_QUEUE | Queue | QUEUE |
|------------------|-------|-------|

| | | |
|------------------------|-------------|-------------|
| EX_CD_SYMB_SUB_QUEUE_1 | Sub-Queue-1 | SUB-QUEUE-1 |
|------------------------|-------------|-------------|

| | | |
|------------------------|-------------|-------------|
| EX_CD_SYMB_SUB_QUEUE_2 | Sub-Queue-2 | SUB-QUEUE-2 |
|------------------------|-------------|-------------|

| | | |
|------------------------|-------------|-------------|
| EX_CD_SYMB_SUB_QUEUE_3 | Sub-Queue-3 | SUB-QUEUE-3 |
|------------------------|-------------|-------------|

| | | |
|-------------------|--------|--------|
| EX_CD_SYMB_SOURCE | Source | SOURCE |
|-------------------|--------|--------|

| | | |
|-----------------|-------------|-------------|
| EX_CD_SYMB_DEST | Destination | DESTINATION |
|-----------------|-------------|-------------|

| | | |
|---------------------|----------|----------|
| EX_CD_SYMB_TERMINAL | Terminal | TERMINAL |
|---------------------|----------|----------|

| | | |
|------------------------------------|---------------------|---------------------|
| EX_CD_MSG_DATE_IS | Message Date Clause | MESSAGE DATE IS \$1 |
| \$1: EX_CD_MSG_DATE_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|------------------------------------|---------------------|---------------------|
| EX_CD_MSG_TIME_IS | Message Time Clause | MESSAGE TIME IS \$1 |
| \$1: EX_CD_MSG_TIME_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|---------------------------------------|--------------------|--------------------|
| EX_CD_TEXT_LENGTH_IS | Text Length Clause | TEXT LENGTH IS \$1 |
| \$1: EX_CD_TEXT_LENGTH_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|-----------------------------------|-------------------|----------------|
| EX_CD_END_KEY_IS | End Key Clause | END KEY IS \$1 |
| \$1: EX_CD_END_KEY_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|--------------------------------------|-------------------|-------------------|
| EX_CD_STATUS_KEY_IS | Status Key Clause | STATUS KEY IS \$1 |
| \$1: EX_CD_STATUS_KEY_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|-------------------------------------|-------------------|------------------|
| EX_CD_ERROR_KEY_IS | Error Key Clause | ERROR KEY IS \$1 |
| \$1: EX_CD_ERROR_KEY_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|-------------------------------------|----------------------|----------------------|
| EX_CD_MSG_COUNT_IS | Message Count Clause | MESSAGE COUNT IS \$1 |
| \$1: EX_CD_MSG_COUNT_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|--------------------------------------|--------------------------|--------------------------|
| EX_CD_DEST_COUNT_IS | Destination Count Clause | DESTINATION COUNT IS \$1 |
| \$1: EX_CD_DEST_COUNT_IS__DEFINED | Data Item Defined | NAMED_OBJ_DEF |

| | | |
|-------------------------|---------------------------------|---|
| EX_CD_DEST_TABLE_OCCURS | Destination Table Occurs Clause | DESTINATION TABLE OCCURS \$1 TIMES{\$2} |
| \$1: CDOCRS_LENGTH | Table Length | INT_NUM_CONST |
| \$2: CDOCRS_INDEXED | Indexed By | DX_INDEXED_BY NULL |

10.5.4 REPORT Section

| | | |
|-------------------------|----------------|--------------------------------|
| SECTION_REPORT | Report Section | REPORT_SECTION.{ \$1? \n \$1 } |
| \$1: SECTION_REPORT_RDS | RD entries | DECL_RD_LIST (DECL_RD) NULL |

| | | |
|--------------|-------------------------|--------|
| DECL_RD_LIST | Report Description List | \r@ \n |
|--------------|-------------------------|--------|

| | | |
|----------------|------------------------------|-------------------------------|
| DECL_RD | Report Description Holder | \r \$1 { \$2? \n \$2 } |
| \$1: RDH_ENTRY | Report Description Entry | DECL_RD_ENTRY |
| \$2: RDH_ARGS | Report Group Decr Entry List | DECL_DD_LIST (DECL_RGD_ENTRY) |

| | | |
|------------------------|---------------------------|--|
| DECL_RD_ENTRY | Report Description Entry | \rRD ~x\$1{\$2?\n\$x\$2}{\$3?\n\$x\$3}{~\$4? |
| \$1: RD_REPORT_NAME | Report Name Defined | NAMED_OBJ_DEF |
| \$2: RD_GLOBAL | Global | DX_GLOBAL NULL |
| \$3: RD_CODE | With Code | EX_RD_CODE NULL |
| \$4: RD_CONTROLS | Controls | EX_RD_CONTROLS NULL |
| \$5: RD_PAGE_LIMITS | Page Limits | EX_RD_PAGE_LIMITS NULL |
| \$6: RD_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$7: RD_LINE_COUNTER | Line-Counter | NAMED_OBJ_DEF |
| \$8: RD_PAGE_COUNTER | Page-Counter | NAMED_OBJ_DEF |

| | | |
|--------------------------|--------------|----------|
| EX_RD_CODE | Code Clause | CODE \$1 |
| \$1: EX_RD_CODE__LITERAL | Code Literal | literal |

| | | |
|---------------------------------|-----------------|--|
| EX_RD_CONTROLS | Controls Clause | CONTROLS ARE \$1 |
| \$1: EX_RD_CONTROLS__AREWHAT | Are What | EX_LIST (EX_RD_FINAL NAMED_OBJ_USE) |

| | | |
|-------------|-------|-------|
| EX_RD_FINAL | Final | FINAL |
|-------------|-------|-------|

| | | |
|---------------------|--------------------|---|
| EX_RD_PAGE_LIMITS | Page Limits Clause | PAGE LIMITS ARE \$1 LINES{\$2? \$2} |
| \$1: RDPGLM_LINES | Lines | INT_NUM_CONST |
| \$2: RDPGLM_PHRASES | Phrases | EX_LIST (EX_RD_HEADING EX_RD_FOOTING EX_RD_FIRST_DETAIL EX_RD_LAST_DETAIL) |

| | | |
|---------------------------|----------------|---------------|
| EX_RD_HEADING | Heading Phrase | HEADING \$1 |
| \$1: EX_RD_HEADING__LINES | Lines | INT_NUM_CONST |

| | | |
|---------------------------|----------------|---------------|
| EX_RD_FOOTING | Footing Phrase | FOOTING \$1 |
| \$1: EX_RD_FOOTING__LINES | Lines | INT_NUM_CONST |

| | | |
|-----------------------------------|---------------------|------------------|
| EX_RD_FIRST_DETAIL | First Detail Phrase | FIRST DETAIL \$1 |
| \$1: EX_RD_FIRST_DETAIL__LINES | Lines | INT_NUM_CONST |

| | | |
|----------------------------------|--------------------|-----------------|
| EX_RD_LAST_DETAIL | Last Detail Phrase | LAST DETAIL \$1 |
| \$1: EX_RD_LAST_DETAIL__LINES | Lines | INT_NUM_CONST |

| | | |
|---------------|---|-------|
| DECL_RGD_LIST | Report Group Description Holder List | \r@\n |
|---------------|---|-------|

| | | |
|-----------------|---------------------------------|-------------------------|
| DECL_RGD | Report Group Description Holder | \r\$1{\$2?\n\t\$2\b} |
| \$1: RGDH_ENTRY | Report Group Description Entry | DECL_RGD_ENTRY |
| \$2: RGDH_ARGS | Subordinate Levels | DECL_DD_LIST (DECL_RGD) |

| | | |
|--------------------------|--------------------------------|--|
| DECL_RGD_ENTRY | Report Group Description Entry | \r\$1 {\$2? \$2}{\$10? \$10}{\$3?\$p3 \$3}{\$3}{\$3} |
| \$1: RGD_LEVEL | Level Number | INT_NUM_CONST |
| \$2: RGD_DATA_NAME | RGD Item Defined | NAMED_OBJ_DEF NULL |
| \$3: RGD_PICTURE | Picture | DX_PICTURE NULL |
| \$4: RGD_USAGE | Usage | DX_USAGE NULL |
| \$5: RGD_BWZ | Blank When Zero | DX_BWZ NULL |
| \$6: RGD_SIGN | Sign | DX_SIGN NULL |
| \$7: RGD_LINE_NUMBER | Line Number | EX_RGD_LINE_NUMBER NULL |
| \$8: RGD_NEXT_GROUP | Next Group | EX_RGD_NEXT_GROUP NULL |
| \$9: RGD_TYPE | Type | EX_RGD_TYPE NULL |
| \$10: RGD_COLUMN_NUMBER | Column Number | EX_RGD_COLUMN_NUMBER NULL |
| \$11: RGD_SOURCE | Source | EX_RGD_SOURCE NULL |
| \$12: RGD_VALUE | Value | EX_RGD_VALUE NULL |
| \$13: RGD_SUM | Sum | EX_RGD_SUM NULL |
| \$14: RGD_GROUP | Group Indicate | EX_RGD_GROUP_INDICATE NULL |
| \$15: RGD_UNINLINED_COPY | Uninlined COPY | copy-expr NULL |

| | | |
|--------------------------------------|--------------------|---|
| EX_RGD_LINE_NUMBER | Line Number Clause | LINE \$1 |
| \$1: EX_RGD_LINE_NUMBER__POSITION | Position | EX_RGD_POS_INT EX_RGD_POS_PLUS EX_RGD_POS_NEXT_PAGE |

| | | |
|----------------------|-----------------|--------------------------------|
| EX_RGD_POS_INT | NN On Next Page | \$1{\$2? \$2} |
| \$1: RGD_LN_LINES | Lines | INT_NUM_CONST |
| \$2: RGD_LN_NEXTPAGE | Next Page | EX_RGD_POS_ON_NEXT_PAGE NULL |

| | | |
|-------------------------|---------------------|--------------|
| EX_RGD_POS_ON_NEXT_PAGE | On Next Page Phrase | ON NEXT PAGE |
|-------------------------|---------------------|--------------|

| | | |
|-----------------------------|----------------|---------------|
| EX_RGD_POS_PLUS | Plus NN Phrase | PLUS \$1 |
| \$1: EX_RGD_POS_PLUS__LINES | Lines | INT_NUM_CONST |

| | | |
|----------------------|------------------|-----------|
| EX_RGD_POS_NEXT_PAGE | Next Page (OSVS) | NEXT PAGE |
|----------------------|------------------|-----------|

| | | |
|--------------------------------------|----------------------|---------------|
| EX_RGD_COLUMN_NUMBER | Column Number Clause | COLUMN \$1 |
| \$1: EX_RGD_COLUMN_NUMBER__NUMBER | Number | INT_NUM_CONST |

| | | |
|-------------------------------------|------------|---|
| EX_RGD_NEXT_GROUP | Next Group | NEXT GROUP \$1 |
| \$1: EX_RGD_NEXT_GROUP__POSITION | Position | EX_RGD_POS_INT EX_RGD_POS_PLUS EX_RGD_POS_NEXT_PAGE |

| | | |
|------------------------|-------------|---|
| EX_RGD_TYPE | Type Clause | TYPE \$1 |
| \$1: EX_RGD_TYPE__TYPE | Type | EX_RGD_REPORT_HEADING EX_RGD_PAGE_HEADING EX_RGD_CONTROL_HEADING EX_RGD_DETAIL EX_RGD_CONTROL_FOOTING EX_RGD_PAGE_FOOTING EX_RGD_REPORT_FOOTING |

| | | |
|-----------------------|-----------------------|----------------|
| EX_RGD_REPORT_HEADING | Report Heading Phrase | REPORT HEADING |
|-----------------------|-----------------------|----------------|

| | | |
|-----------------------|-----------------------|----------------|
| EX_RGD_REPORT_FOOTING | Report Footing Phrase | REPORT FOOTING |
|-----------------------|-----------------------|----------------|

| | | |
|---------------------|---------------------|--------------|
| EX_RGD_PAGE_HEADING | Page Heading Phrase | PAGE HEADING |
|---------------------|---------------------|--------------|

| | | |
|---------------------|---------------------|--------------|
| EX_RGD_PAGE_FOOTING | Page Footing Phrase | PAGE FOOTING |
|---------------------|---------------------|--------------|

| | | |
|--|------------------------|--------------------------------|
| EX_RGD_CONTROL_HEADING | Control Heading Phrase | CONTROL HEADING \$1 |
| \$1: EX_RGD_CONTROL_HEADING__REPORT | Report Name | NAMED_OBJ_USE EX_RD_FINAL |

| | | |
|--|------------------------|--------------------------------|
| EX_RGD_CONTROL_FOOTING | Control Footing Phrase | CONTROL FOOTING \$1 |
| \$1: EX_RGD_CONTROL_FOOTING__REPORT | Report Name | NAMED_OBJ_USE EX_RD_FINAL |

| | | |
|---------------|---------------|--------|
| EX_RGD_DETAIL | Detail Phrase | DETAIL |
|---------------|---------------|--------|

| | | |
|---------------------------|---------------|------------|
| EX_RGD_SOURCE | Source Clause | SOURCE \$1 |
| \$1: EX_RGD_SOURCE__IDENT | Source | identifier |

| | | |
|----------------------------|--------------|-----------|
| EX_RGD_VALUE | Value Clause | VALUE \$1 |
| \$1: EX_RGD_VALUE__LITERAL | Literal | literal |

| | | |
|---------------------|-------------|-----------------------------|
| EX_RGD_SUM | Sum Clause | SUM \$1{\$2? \$2}{\$3? \$3} |
| \$1: RGD_SUM_IDENTS | Identifiers | IDENT_LIST (identifier) |
| \$2: RGD_SUM_UPON | Upon | EX_RGD_SUM_UPON NULL |
| \$3: RGD_SUM_RESET | Reset | EX_RGD_SUM_RESET_ON NULL |

| | | |
|------------------------------------|---------------------|-------------------------|
| EX_RGD_SUM_UPON | RGD Sum Upon Phrase | UPON \$1 |
| \$1: EX_RGD_SUM_UPON__DATAITEMS | Data Items | EX_LIST (NAMED_OBJ_USE) |

| | | |
|-------------------------------------|----------------------|--------------------------------|
| EX_RGD_SUM_RESET_ON | RGD Sum Reset Phrase | RESET ON \$1 |
| \$1: EX_RGD_SUM_RESET_ON__REPORT | Reset On Report | NAMED_OBJ_USE EX_RD_FINAL |

| | | |
|-----------------------|----------------|-------|
| EX_RGD_GROUP_INDICATE | Group Indicate | GROUP |
|-----------------------|----------------|-------|

10.5.5 SCREEN Section

| | | |
|--------------------------------|------------------------|---------------------------------|
| SECTION_SCREEN | Screen Section | SCREEN_SECTION.{ \$1? \n \$1 } |
| \$1: SECTION_SCREEN__DESCRS | Screen Descriptor List | DECL_DD_LIST (DECL_SCRD) NULL |

| | | |
|------------------|---------------------------|---------------------------------|
| DECL_SCRD | Screen Description Holder | \r \$1 { \$2? \n \t \$2 \b } |
| \$1: SCRDH_ENTRY | Screen Description Entry | DECL_SCRD_ENTRY |
| \$2: SCRDH_ARGS | Subordinate levels | DECL_DD_LIST (DECL_SCRD) NULL |

| | | |
|-----------------------------------|--------------------------|---|
| DECL_SCRD_ENTRY | Screen Description Entry | \$1 ^x~{\$2? \$x\$2}{\$3? \$x\$3}{\$4? \$x\$4} |
| \$1: IX_ADIS_LEVEL | Level | INT_NUM_CONST |
| \$2: IX_ADIS_DATA_NAME | Screen Item Defined | NAMED_OBJ_DEF EX_FILLER NULL |
| \$3: IX_ADIS_LINE_NUMBER | Line Number | SCRD_LINE_NUMBER NULL |
| \$4: IX_ADIS_COLUMN_NUMBER | Column Number | SCRD_COLUMN_NUMBER NULL |
| \$5: IX_ADIS_AT_POS | At | DSPL_AT NULL |
| \$6: IX_ADIS_FROM_UPON | From or Upon | ACCP_FROM DSPL_UPON DSPL_UNIT NULL |
| \$7: IX_ADIS_MODE_IS_BLOCK | Mode Is Block | DSPL_MODE_IS_BLOCK NULL |
| \$8: IX_ADIS_WITH | With Word | DSPL_WITH NULL |
| \$9: IX_ADIS_BELL | Bell | DSPL_BELL NULL |
| \$10: IX_ADIS_BLINK | Blink | DSPL_BLINK NULL |
| \$11: IX_ADIS_ERASE | Erase | DSPL_ERASE NULL |
| \$12: IX_ADIS_GRID | Grid | DSPL_GRID NULL |
| \$13: IX_ADIS_HIGH_LOW | Highlight/Lowlight | DSPL_HIGHLIGHT DSPL_HIGH DSPL_LOWLIGHT DSPL_LOW NULL |
| \$14: IX_ADIS_LEFTLINE | Leftline | DSPL_LEFTLINE NULL |
| \$15: IX_ADIS_OVERLINE | Overline | DSPL_OVERLINE NULL |
| \$16: IX_ADIS_REVERSE | Reverse-Video | DSPL_REVERSE_VIDEO DSPL_REVERSE NULL |
| \$17: IX_ADIS_SIZE | Size | DSPL_SIZE NULL |
| \$18: IX_ADIS_UNDERLINE | Underline | DSPL_UNDERLINE NULL |
| \$19: IX_ADIS_FOREGROUND_COLOR | Foreground-Color | DSPL_FOREGROUND_COLOR NULL |
| \$20: IX_ADIS_BACKGROUND_COLOR | Background-Color | DSPL_FOREGROUND_COLOR NULL |
| \$21: IX_ADIS_CONTROL | Control | DSPL_CONTROL NULL |

| | | |
|---------------------------------|---------------------------|--|
| DECL_SCRD_ENTRY | (continued) | (continued) |
| \$22: IX_ADIS_PROMPT | Prompt | SCRD_PROMPT ACCP_PROMPT NULL |
| \$23: IX_ADIS_AUTO | Auto | ACCP_AUTO NULL |
| \$24: IX_ADIS_FULL | Full | ACCP_FULL NULL |
| \$25: IX_ADIS_REQUIRED | Required | ACCP_REQUIRED NULL |
| \$26: IX_ADIS_SECURE | Secure | ACCP_SECURE NULL |
| \$27: IX_ADIS_ZERO_FILL | Zero-Fill | ACCP_ZERO_FILL NULL |
| \$28: IX_ADIS_JUST | Just | SCRD_JUSTIFIED ACCP_LEFT_JUSTIFY ACCP_RIGHT_JUSTIFY NULL |
| \$29: IX_ADIS_SIGN | Sign | SCRD_SIGN ACCP_TRAILING_SIGN NULL |
| \$30: IX_ADIS_BLANK | Blank | DSPL_BLANK NULL |
| \$31: IX_ADIS_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$32: IX_SCRD_VALUE | Value | SCRD_VALUE NULL |
| \$33: IX_SCRD_PICTURE | Picture | SCRD_PICTURE NULL |
| \$34: IX_SCRD_FROM | From | SCRD_FROM NULL |
| \$35: IX_SCRD_TO | To | SCRD_TO NULL |
| \$36: IX_SCRD_USING | Using | SCRD_USING NULL |
| \$37: IX_SCRD_USAGE | Usage | SCRD_USAGE NULL |
| \$38: IX_SCRD_BWZ | Bwz | SCRD_BWZ NULL |
| \$39: IX_SCRD_OCCURS | Occurs | SCRD_OCCURS NULL |

| | | |
|-----------|-------------|------|
| DSPL_BELL | Bell Clause | BELL |
|-----------|-------------|------|

| | | |
|------------|--------------|-------|
| DSPL_BLINK | Blink Clause | BLINK |
|------------|--------------|-------|

| | | |
|----------------------|--------------|-------------------------------|
| DSPL_ERASE | Erase Clause | ERASE{\$1? \$1} |
| \$1: DSPL_ERASE_EOLS | EOL/EOS | DSPL_EOL DSPL_EOS NULL |

| | | |
|----------|------------|-----|
| DSPL_EOL | EOL Phrase | EOL |
|----------|------------|-----|

| | | |
|----------|------------|-----|
| DSPL_EOS | EOS Phrase | EOS |
|----------|------------|-----|

| | | |
|-----------|-------------|------|
| DSPL_GRID | Grid Clause | GRID |
|-----------|-------------|------|

| | | |
|----------------|------------------|-----------|
| DSPL_HIGHLIGHT | Highlight Clause | HIGHLIGHT |
|----------------|------------------|-----------|

| | | |
|--------------------------------------|-------------------------|----------------------|
| DSPL_LOWLIGHT | Lowlight Clause | LOWLIGHT |
| DSPL_LEFTLINE | Leftline Clause | LEFTLINE |
| DSPL_OVERLINE | Overline Clause | OVERLINE |
| DSPL_REVERSE_VIDEO | Reverse-Video Clause | REVERSE-VIDEO |
| DSPL_UNDERLINE | Underline Clause | UNDERLINE |
| DSPL_SIZE | Size Clause | SIZE \$1 |
| \$1: DSPL_SIZE__VALUE | Size | ident-liter |
| DSPL_FOREGROUND_COLOR | Foreground Color Clause | BACKGROUND-COLOR \$1 |
| \$1: DSPL_FOREGROUND_COLOR__VALUE | Color Value | ident-liter |
| DSPL_BACKGROUND_COLOR | Background Color Clause | BACKGROUND-COLOR \$1 |
| \$1: DSPL_BACKGROUND_COLOR__VALUE | Color Value | ident-liter |
| DSPL_CONTROL | Control Clause | CONTROL \$1 |
| \$1: DSPL_CONTROL__VALUE | Control Value | ident-liter |
| ACCP_AUTO | Auto Clause | AUTO |
| ACCP_FULL | Full Clause | FULL |
| ACCP_REQUIRED | Required Clause | REQUIRED |
| ACCP_SECURE | Secure Clause | SECURE |
| ACCP_ZERO_FILL | Zero Clause | ZERO-FILL |

| | | |
|---------------------|--------------|----------------------------|
| DSPL_BLANK | Blank Clause | BLANK \$1 |
| \$1: DSPL_BLANK__SL | Screen/Line | DSPL_SCREEN DSPL_LINE |

| | | |
|-------------|---------------|--------|
| DSPL_SCREEN | Screen Clause | SCREEN |
|-------------|---------------|--------|

| | | |
|-----------|-------------|------|
| DSPL_LINE | Line Clause | LINE |
|-----------|-------------|------|

| | | |
|-------------------------|-------------------|---------------|
| SCRD_PROMPT | Prompt Clause | PROMPT IS \$1 |
| \$1: SCRD_PROMPT__VALUE | Prompt Char Value | ident-liter |

| | | |
|------------------------|-------------|----------------------------------|
| SCRD_LINE_NUMBER | Line Clause | LINE {\$1?\$1 }{\$2?\$2} |
| \$1: ADIS_LINENO_PM | Plus/Minus | SCRD_PLUS SCRD_MINUS NULL |
| \$2: ADIS_LINENO_VALUE | Value | ident-liter |

| | | |
|-----------------------|---------------|----------------------------------|
| SCRD_COLUMN_NUMBER | Column Clause | COLUMN {\$1?\$1 }{\$2?\$2} |
| \$1: ADIS_COLNO_PM | Plus/Minus | SCRD_PLUS SCRD_MINUS NULL |
| \$2: ADIS_COLNO_VALUE | Value | ident-liter |

| | | |
|-----------|------|---|
| SCRD_PLUS | Plus | + |
|-----------|------|---|

| | | |
|------------|-------|---|
| SCRD_MINUS | Minus | - |
|------------|-------|---|

| | | |
|---------------------------|-----------------|-------------|
| SCRD_PICTURE | Pic Clause | PIC \$1 |
| \$1: SCRD_PICTURE__STRING | Pic Char String | CHAR_STRING |

| | | |
|-----------------------|-------------|-------------|
| SCRD_FROM | From Clause | FROM \$1 |
| \$1: SCRD_FROM__VALUE | From Value | ident-liter |

| | | |
|---------------------|--------------|------------|
| SCRD_TO | To Clause | TO \$1 |
| \$1: SCRD_TO__IDENT | To Data Item | identifier |

| | | |
|------------------------|-----------------|------------|
| SCRD_USING | Using Clause | USING \$1 |
| \$1: SCRD_USING__IDENT | Using Data Item | identifier |

| | | |
|------------------------|--------------|-----------|
| SCRD_VALUE | Value Clause | VALUE \$1 |
| \$1: SCRD_VALUE__VALUE | Value | literal |

| | | |
|-----------------------|--------------|------------|
| SCRD_USAGE | Usage Clause | USAGE \$1 |
| \$1: SCRD_USAGE__TYPE | Usage | DU_DISPLAY |

| | | |
|----------|--------------|-----------------|
| SCRD_BWZ | Blank Clause | BLANK WHEN ZERO |
|----------|--------------|-----------------|

| | | |
|----------------|------------------|-----------|
| SCRD_JUSTIFIED | Justified Clause | JUSTIFIED |
|----------------|------------------|-----------|

| | | |
|---------------------|------------------|---------------------------------------|
| SCRD_SIGN | Sign Clause | SIGN IS \$1{\$2? \$2} |
| \$1: SCRD_SIGN__LT | Leading/Trailing | DX_SIGN_LEADING DX_SIGN_TRAILING |
| \$2: SCRD_SIGN__SEP | Separate | DX_SIGN_SEPARATE NULL |

| | | |
|--------------------------|----------------------|--------------------|
| SCRD_OCCURS | Occurs Clause | OCCURS \$1[TIMES] |
| \$1: SCRD_OCCURS__NUMBER | Number Of Occurences | INT_NUM_CONST |

10.6 PROCEDURE Division

| | | |
|----------------------|-------------------------------|--|
| DIVISION_PROCEDURE | Procedure Division | PROCEDURE DIVISION{\$1? \$1}{\$2? \$2}{\$3? \$3} |
| \$1: PD_MNEMONIC | Call Convention Mnemonic (MF) | NAMED_OBJ_USE NULL |
| \$2: PD_LINKAGE | Call Linkage (FSC) | EX_CALL_WITH_LINKAGE NULL |
| \$3: PD_USING | Using Clause | EX_PROC_USING EX_PROC_CHAINING NULL |
| \$4: PD_GIVING | Giving Clause | EX_CALL_GIVING NULL |
| \$5: PD_DECLARATIVES | Declaratives | DECLARATIVE_SECTIONS NULL |
| \$6: PD_SECTIONS | Imperatives | SECTION_PROC_LIST (SECTION_PROC) NULL |

| | | |
|------------------------------------|---------------------|------------------|
| EX_CALL_WITH_LINKAGE | With Linkage Clause | WITH \$1 LINKAGE |
| \$1: EX_CALL_WITH_LINKAGE__TYPE | Linkage Type | CHAR_STRING |

| | | |
|----------------------------|---------------------|---------------------------------------|
| EX_PROC_USING | Callee Using Clause | USING \$1 |
| \$1: EX_PROC_USING__PARAMS | Parameter List | EX_PROC_PARAM_LIST (EX_PROC_PARAM) |

| | | |
|----------------------------------|------------------------|---------------------------------------|
| EX_PROC_CHAINING | Callee Chaining Clause | CHAINING \$1 |
| \$1: EX_PROC_CHAINING__PARAMS | Parameter List | EX_PROC_PARAM_LIST (EX_PROC_PARAM) |

| | | |
|--------------------|---------------------------|-----|
| EX_PROC_PARAM_LIST | Proceduure Parameter List | @\n |
|--------------------|---------------------------|-----|

| | | |
|-----------------------------------|-----------------------|---|
| EX_PROC_PARAM | Procedure Parameter | { $\$1?$ $\$1$ }{ $\$2=\2 } |
| $\$1$: PROC_PARAM_TYPE_PHRASE | Parameter Type Phrase | EX_PT_BY_REFERENCE EX_PT_BY_VALUE NULL |
| $\$2$: PROC_PARAM_ARG | Parameter Data Item | NAMED_OBJ_USE |
| $\$3$: PROC_PARAM_TYPE | Actual Parameter Type | EX_PT_BY_REFERENCE EX_PT_BY_VALUE |

10.6.1 General Division Structure

| | | |
|------------------------|----------------------|---|
| DECLARATIVE_SECTIONS | Declarative Sections | DECLARATIVES.\r{ $\$1?$ \n $\$1$ }{ $\$2?$ \n $\$2$ } |
| $\$1$: DECLS_SECTIONS | Sentences | SECTION_PROC_LIST (SECTION_PROC) NULL |
| $\$2$: DECLS_END | End-Declaratives | END_DECLARATIVES NULL |

| | | |
|------------------|-------------------------|-------------------|
| END_DECLARATIVES | End-Declaratives Header | END DECLARATIVES. |
|------------------|-------------------------|-------------------|

| | | |
|-------------------|--------------|-------|
| SECTION_PROC_LIST | Section List | \r@\n |
|-------------------|--------------|-------|

| | | |
|-------------------------|----------------------|--|
| SECTION_PROC | Section | $\$1$ SECTION{ $\$2?$ $\$2$ }\r{ $\$3?$ \n $\$3$ } |
| $\$1$: SECTION_NAME | Section Name Defined | NAMED_OBJ_DEF NULL |
| $\$2$: SECTION_SEGMENT | Segment Number | INT_NUM_CONST NULL |
| $\$3$: SECTION_PARAS | Paragraphs | PARA_PROC_LIST (PARA_PROC) NULL |

| | | |
|----------------|----------------|-------|
| PARA_PROC_LIST | Paragraph List | \r@\n |
|----------------|----------------|-------|

| | | |
|------------------------|-------------------|--|
| PARA_PROC | Paragraph | $\$1$.\r{ $\$2?$ \n $\$2$ } |
| $\$1$: PARA_NAME | Para Name Defined | NAMED_OBJ_DEF NULL |
| $\$2$: PARA_SENTENCES | Sentences | SENTENCE_LIST (SENTENCE SENTENCE_EMPTY) |

| | | |
|---------------|---------------|-------|
| SENTENCE_LIST | Sentence List | \r@\n |
|---------------|---------------|-------|

| | | |
|------------------------|-------------------|--------------------------|
| SENTENCE | Sentence | \r{ $\$1?$ \t $\$1$.\b} |
| $\$1$: SENTENCE_STMTS | Para Name Defined | STMT_LIST (statement) |

| | | |
|----------------|----------------|---|
| SENTENCE_EMPTY | Empty Sentence | . |
|----------------|----------------|---|

| | | |
|-----------|----------------|-------|
| STMT_LIST | Statement List | \r@\n |
|-----------|----------------|-------|

| | | |
|--------------------|-------------------|--------------------|
| STMT__SYNTAX_ERROR | Syntax Error stmt | \\[syntax-error\\] |
|--------------------|-------------------|--------------------|

| | | |
|-------------|-----------|------------------------------|
| • statement | Statement | statement-1 statement-2 |
|-------------|-----------|------------------------------|

| | | |
|---------------|-----------|--|
| • statement-1 | Statement | copy-expr STMT_NO_OP STMT_REPLACE STMT_ACCEPT STMT_ACCEPT_MSG_COUNT STMT_ADD STMT_ALTER STMT_CALL STMT_CANCEL STMT_CHAIN STMT_CLOSE STMT_COMMIT STMT_COMPUTE STMT_CONTINUE STMT_DELETE STMT_DELETE_FILE STMT_DISABLE STMT_DISPLAY STMT_DIVIDE STMT_ENABLE STMT_ENTRY STMT_EVALUATE STMT_EXAMINE_TALLYING STMT_EXAMINE_REPLACING STMT_EXEC_IN_DCL STMT_EXEC STMT_EXHIBIT STMT_EXIT STMT_EXIT_PROGRAM STMT_EXIT_PERFORM STMT_EXIT_PARAGRAPH STMT_EXIT_SECTION STMT_GENERATE STMT_GOTO STMT_GOTO_DEPENDING STMT_GOBACK |
|---------------|-----------|--|

| | | |
|---------------|---------------------|---|
| • statement-2 | Statement continued | STMT_IF STMT_INITIALIZE STMT_INITIATE STMT_INSPECT STMT_INSPECT_CONVERTING STMT_MERGE STMT_MOVE STMT_MOVE_CORR STMT_MULTIPLY STMT_NEXT_SENTENCE STMT_NOTE STMT_OPEN STMT_PERFORM_PROC STMT_PERFORM_BLOCK STMT_PURGE STMT_READ STMT_READY_TRACE STMT_RECEIVE STMT_RELEASE STMT_RESET_TRACE STMT_RETURN STMT_REWRITE STMT_ROLLBACK STMT_SEARCH STMT_SEARCH_ALL STMT_SEND STMT_SERVICE_RELOAD STMT_SET_TO STMT_SET_UP_DOWN_BY STMT_SET_ADDRESS STMT_SORT STMT_START STMT_STOP STMT_STRING STMT_SUBTRACT STMT_SUPPRESS_PRINTING STMT_TERMINATE STMT_TRANSFORM STMT_UNLOCK STMT_UNDELETE STMT_UNSTRING STMT_USE_AFTER STMT_USE_AFTER_DEADLOCK STMT_USE_BEFORE_REPORTING STMT_USE_FOR_DEBUGGING STMT_WRITE |
|---------------|---------------------|---|

| | | |
|------------|------------|------|
| STMT_NO_OP | No-Op Stmt | \377 |
|------------|------------|------|

10.6.2 Copy Statements

| | | |
|-------------|---------------------------|---|
| • copy-expr | CopyBegin/End/SyntaxError | STMT_UNINLINED_COPY STMT_INLINED_COPY_BEG STMT_INLINED_COPY_END STMT_INLINED_INCLUDE_BEG STMT_INLINED_INCLUDE_END STMT_REPLACE STMT_REPLACE_OFF STMT__SYNTAX_ERROR |
|-------------|---------------------------|---|

| | | |
|------------------------|----------------|---|
| STMT_UNINLINED_COPY | UnInlined Copy | COPY{\$1? \$1}{\$2? \$2}[.] |
| \$1: AX_COPY_FILE | File Name | OF_IN_LIST (STR_CONST CHAR_STRING) |
| \$2: AX_COPY_REPLACING | Replacing | COPY_REPLACING NULL |
| \$3: AX_COPY_REAL_FILE | Real File Name | CHAR_STRING |
| \$4: AX_COPY_LEVEL | Nesting Level | INT_NUM_CONST |

| | | |
|--|--------------------|---|
| STMT_INLINED_COPY_BEG | Inlined Copy Begin | COPY{\$1? \$1}{\$2? \$2}[.] |
| \$1: STMT_INLINED_COPY_BEG__FILE | File Name | OF_IN_LIST (STR_CONST CHAR_STRING) |
| \$2: STMT_INLINED_COPY_BEG__REPLACING | Replacing | COPY_REPLACING NULL |
| \$3: STMT_INLINED_COPY_BEG__REAL_FILE | Real File Name | CHAR_STRING |
| \$4: STMT_INLINED_COPY_BEG__LEVEL | Nesting Level | INT_NUM_CONST |

| | | |
|-----------------------|------------------|---------------|
| STMT_INLINED_COPY_END | Inlined Copy End | |
| \$1: AX_EOF_REAL_FILE | Real File Name | CHAR_STRING |
| \$2: AX_EOF_LEVEL | Nesting Level | INT_NUM_CONST |

| | | |
|---------------------------------|------------------|---|
| COPY_REPLACING | Replacing Clause | REPLACING \$1 |
| \$1: COPY_REPLACING__PHRASES | Phrases | COPY_REPLACING_LIST (COPY_REPLACING_BY COPY_REPLACING_BY_ALIGNED) |

| | | |
|---------------------|--------------------------|-----|
| COPY_REPLACING_LIST | Replacing By Phrase List | @\n |
|---------------------|--------------------------|-----|

| | | |
|---------------------|---------------------|---|
| COPY_REPLACING_BY | Replacing By Phrase | \$1{\$2= BY \$2} |
| \$1: AX_REPLBY_FROM | Replacing What | COPY_EQEQS COPY_TOKEN_LIST (CHAR_STRING) |
| \$2: AX_REPLBY_TO | Replacing By What | COPY_EQEQS COPY_TOKEN_LIST (CHAR_STRING) |

| | | |
|---|----------------------------------|---|
| COPY_REPLACING_BY_ALIGNED | Replacing By Phrase (Aligned) | \$1{\$2=\$p4 BY \$2} |
| \$1: COPY_REPLACING_BY_ALIGNED__FROM | Replacing What | COPY_EQEQS COPY_TOKEN_LIST (CHAR_STRING) |
| \$2: COPY_REPLACING_BY_ALIGNED__TO | Replacing By What | COPY_EQEQS COPY_TOKEN_LIST (CHAR_STRING) |

| | | |
|-------------------------|----------------|-------------------------------|
| COPY_EQEQS | Brackets == == | ==\$1== |
| \$1: COPY_EQEQS__TOKENS | Tokens | COPY_TOKEN_LIST (CHAR_STRING) |

| | | |
|-----------------|-----------------|---|
| COPY_TOKEN_LIST | Copy Token List | @ |
|-----------------|-----------------|---|

| | | |
|----------------------------|-----------------|--|
| STMT_REPLACE | Replace Stmt | REPLACE{\$1?\n\$1}[.] |
| \$1: STMT_REPLACE__PHRASES | Replace Phrases | COPY_REPLACING_LIST (COPY_REPLACING_BY) |

| | | |
|------------------|------------------|--------------|
| STMT_REPLACE_OFF | Replace Off Stmt | REPLACE OFF. |
|------------------|------------------|--------------|

10.6.3 ACCEPT statement

| | | |
|----------------------------|------------------|--|
| STMT_ACCEPT | Accept Statement | ACCEPT \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: STMT_ACCEPT__ITEMS | Accept Item List | ACCEPT_ITEM_LIST (ACCEPT_FUNC_ITEM ACCEPT_ANSI_ITEM ACCEPT_SCREEN_ITEM) |
| \$2: STMT_ACCEPT__ONEXC | On Exception | ON_ACCEPT_EXCEPTION NULL |
| \$3: STMT_ACCEPT__NOTONEXC | Not On Exception | NOT_ON_ACCEPT_EXCEPTION NULL |
| \$4: STMT_ACCEPT__END | End-Accept | EX_END_ACCEPT NULL |

| | | |
|------------------|------------------|-----|
| ACCEPT_ITEM_LIST | Accept Item List | @\n |
|------------------|------------------|-----|

| | | |
|-------------------------------------|---------------------------|---------------------------------------|
| ACCEPT_FUNC_ITEM | ANSI Accept Item | { \$1=\$1 } { \$2? \$2 } { \$3? \$3 } |
| \$1: IX_ACCP_FUNC_DATA_NAME | To Identifier | identifier |
| \$2: IX_ACCP_FUNC_FROM | From What | ACCP_FROM NULL |
| \$3: IX_ACCP_FUNC_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|-------------------------------------|---------------------------|---------------------------------------|
| ACCEPT_ANSI_ITEM | ANSI Accept Item | { \$1=\$1 } { \$2? \$2 } { \$3? \$3 } |
| \$1: IX_ACCP_ANSI_DATA_NAME | To Identifier | identifier |
| \$2: IX_ACCP_ANSI_FROM | From What | ACCP_FROM NULL |
| \$3: IX_ACCP_ANSI_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|-----------------------------------|--------------------|---|
| ACCEPT_SCREEN_ITEM | Screen Accept Item | { \$1? \$1 } { \$2 = \$2 } { \$3 ? \$3 } { \$4 ? \$4 } { \$5 ? |
| \$1: IX_ACCP_POSN_SPEC | Position Spec (MS) | EX_ADIS_MS_POSN NULL |
| \$2: IX_ACCP_DATA_NAME | Accept To | identifier |
| \$3: IX_ACCP_LINE_NUMBER | Line Number | DSPL_LINE_NUMBER NULL |
| \$4: IX_ACCP_COLUMN_NUMBER | Column Number | DSPL_COLUMN_NUMBER NULL |
| \$5: IX_ACCP_AT_POS | At | DSPL_AT DSPL_POSITION NULL |
| \$6: IX_ACCP_FROM_UPON | From or Upon | ACCP_FROM DSPL_UNIT NULL |
| \$7: IX_ACCP_MODE_IS_BLOCK | Mode Is Block | DSPL_MODE_IS_BLOCK NULL |
| \$8: IX_ACCP_WITH | With Word | DSPL_WITH NULL |
| \$9: IX_ACCP_BELL | Bell | DSPL_BELL ACCP_NO_BEEP NULL |
| \$10: IX_ACCP_BLINK | Blink | DSPL_BLINK NULL |
| \$11: IX_ACCP_ERASE | Erase | DSPL_ERASE NULL |
| \$12: IX_ACCP_GRID | Grid | DSPL_GRID NULL |
| \$13: IX_ACCP_HIGH_LOW | Highlight/Lowlight | DSPL_HIGHLIGHT DSPL_HIGH DSPL_LOWLIGHT DSPL_LOW NULL |
| \$14: IX_ACCP_LEFTLINE | Leftline | DSPL_LOWLIGHT NULL |
| \$15: IX_ACCP_OVERLINE | Overline | DSPL_OVERLINE NULL |
| \$16: IX_ACCP_REVERSE | Reverse-Video | DSPL_REVERSE_VIDEO DSPL_REVERSE NULL |
| \$17: IX_ACCP_SIZE | Size | DSPL_SIZE NULL |
| \$18: IX_ACCP_UNDERLINE | Underline | DSPL_UNDERLINE NULL |
| \$19: IX_ACCP_FOREGROUND_COLOR | Foreground-Color | DSPL_FOREGROUND_COLOR NULL |
| \$20: IX_ACCP_BACKGROUND_COLOR | Background-Color | DSPL_FOREGROUND_COLOR NULL |
| \$21: IX_ACCP_CONTROL | Control | DSPL_CONTROL NULL |

| | | |
|---------------------------------|---------------------------|--|
| ACCEPT_SCREEN_ITEM | (continued) | (continued) |
| \$22: IX_ACCP_PROMPT | Prompt | SCRD_PROMPT ACCP_PROMPT NULL |
| \$23: IX_ACCP_AUTO | Auto | ACCP_AUTO NULL |
| \$24: IX_ACCP_FULL | Full | ACCP_FULL NULL |
| \$25: IX_ACCP_REQUIRED | Required | ACCP_REQUIRED NULL |
| \$26: IX_ACCP_SECURE | Secure | ACCP_SECURE ACCP_ECHO NULL |
| \$27: IX_ACCP_ZERO_FILL | Zero-Fill | ACCP_ZERO_FILL NULL |
| \$28: IX_ACCP_JUST | Just | SCRD_JUSTIFIED ACCP_LEFT_JUSTIFY ACCP_RIGHT_JUSTIFY NULL |
| \$29: IX_ACCP_SIGN | Sign | SCRD_SIGN ACCP_TRAILING_SIGN NULL |
| \$30: IX_ACCP_BLANK | Blank | DSPL_BLANK NULL |
| \$31: IX_ACCP_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$32: IX_ACCP_CONVERT | Convert | ACCP_CONVERT NULL |
| \$33: IX_ACCP_TAB | Tab | ACCP_TAB NULL |
| \$34: IX_ACCP_OFF | Off | ACCP_OFF NULL |
| \$35: IX_ACCP_TIMEOUT | Timeout | ACCP_TIMEOUT NULL |
| \$36: IX_ACCP_SPACE_FILL | Fill | ACCP_SPACE_FILL NULL |
| \$37: IX_ACCP_UPDATE | Update | ACCP_UPDATE NULL |
| \$38: IX_ACCP_UPPER | Upper | ACCP_UPPER NULL |
| \$39: IX_ACCP_LOWER | Lower | ACCP_LOWER NULL |

| | | |
|--------------------|------------------------------|---|
| EX_ADIS_MS_POSN | Display Position Clause (MS) | (\$1,{ \$2= \$2}[]) |
| \$1: ACCP_MSP_VERT | Vertical | ident-liter OP_ADD_IX OP_SUB_IX |
| \$2: ACCP_MSP_HOR | Horizontal | ident-liter OP_ADD_IX OP_SUB_IX |

| | | |
|------------------------|--------------------|--|
| ACCP_FROM | Accept From Clause | FROM \$1 |
| \$1: ACCP_FROM__SOURCE | Source | io-function ACCP_DATE ACCP_DAY ACCP_DAY_OF_WEEK ACCP_TIME ACCP_LINE_NUMBER ACCP_USER_NAME ACCP_ESC_KEY ACCP_EXC_STATUS |

| | | |
|-----------|------|------|
| ACCP_DATE | Date | DATE |
|-----------|------|------|

| | | |
|---------------------------|------------------|------------------|
| ACCP_DAY | Day | DAY |
| ACCP_DAY_OF_WEEK | Day Of Week | DAY-OF-WEEK |
| ACCP_TIME | Time | TIME |
| ACCP_ENVIRONMENT | Environment | ENVIRONMENT |
| ACCP_LINE_NUMBER | Line Number | LINE NUMBER |
| ACCP_USER_NAME | User Name | USER NAME |
| ACCP_ESC_KEY | Escape Key | ESCAPE KEY |
| ACCP_EXC_STATUS | Exception Status | EXCEPTION STATUS |
| ACCP_NO_BEEP | No Beep | NO BEEP |
| ACCP_PROMPT | Prompt Clause | PROMPT \$1 |
| \$1: ACCP_PROMPT__LITERAL | Prompt Literal | literal NULL |
| ACCP_LEFT_JUSTIFY | Left-Justify | LEFT-JUSTIFY |
| ACCP_RIGHT_JUSTIFY | Right-Justify | RIGHT-JUSTIFY |
| ACCP_SPACE_FILL | Space-Fill | SPACE-FILL |
| ACCP_TRAILING_SIGN | Trailing-Sign | TRAILING-SIGN |
| ACCP_UPDATE | Update | UPDATE |
| ACCP_ECHO | Echo | ECHO |

| | | |
|--------------|---------|---------|
| ACCP_CONVERT | Convert | CONVERT |
|--------------|---------|---------|

| | | |
|----------|-----|-----|
| ACCP_TAB | Tab | TAB |
|----------|-----|-----|

| | | |
|----------|-----|-----|
| ACCP_OFF | Off | OFF |
|----------|-----|-----|

| | | |
|-------------------------|----------------------|-------------------|
| ACCP_TIMEOUT | Timeout After Phrase | TIMEOUT AFTER \$1 |
| \$1: ACCP_TIMEOUT__TIME | Time | ident-liter |

| | | |
|------------|-------|-------|
| ACCP_UPPER | Upper | UPPER |
|------------|-------|-------|

| | | |
|------------|-------|-------|
| ACCP_LOWER | Lower | LOWER |
|------------|-------|-------|

| | | |
|----------------------|--------------------------|---------------------------------------|
| ON_ACCEPT_EXCEPTION | On Accept Exception Stmt | ON EXCEPTION{\$1? \$1}{\$2?\n\t\$2\b} |
| \$1: ONACCPEXC_IDENT | Identifier (RM) | identifier |
| \$2: ONACCPEXC_STMTS | Statements | STMT_LIST (statement) |

| | | |
|-------------------------|------------------------------|---|
| NOT_ON_ACCEPT_EXCEPTION | Not On Accept Exception Stmt | NOT ON EXCEPTION{\$1? \$1}{\$2?\n\t\$2\b} |
| \$1: NOTONACCPEXC_IDENT | Identifier (RM) | identifier |
| \$2: NOTONACCPEXC_STMTS | Statements | STMT_LIST (statement) |

| | | |
|---------------|------------|------------|
| EX_END_ACCEPT | End-Accept | END-ACCEPT |
|---------------|------------|------------|

| | | |
|--------------------------------|---------------------------|----------------------------|
| STMT_ACCEPT_MSG_COUNT | Accept Message Count Stmt | ACCEPT \$1[MESSAGE COUNT] |
| \$1: STMT_ACCEPT_MSG_COUNT__CD | CD name | NAMED_OBJ_USE |

10.6.4 ADD statement

| | | |
|--------------------|-------------------|--|
| STMT_ADD | Add Statement | ADD \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: ADD_FORMAT | Add Format | EX_ARITH_ADD_TO EX_ARITH_ADD_GIVING EX_ARITH_ADD_CORR_TO |
| \$2: ADD_ONSIZE | On Size Error | EX_ON_SIZE_ERROR NULL |
| \$3: ADD_NOTONSIZE | Not On Size Error | EX_NOT_ON_SIZE_ERROR NULL |
| \$4: ADD_END_ADD | End-Add | EX_END_ADD NULL |

| | | |
|-------------------------------------|--------------------------|----------------------------------|
| EX_ARITH_ADD_TO | Add To Format | \$1{\$2= TO \$2} |
| \$1: ADD.ATO_FROM | Add Sum Of | EX_LIST (ident-liter) |
| \$2: ADD.ATO_TO | Add To Every Of | EX_LIST (identifier-rounded) |
| EX_ARITH_ADD_GIVING | Add Giving Format | \$1{\$2= GIVING \$2} |
| \$1: ADD.AGIVE_FROM | Add Sum Of | EX_LIST (ident-liter) |
| \$2: ADD.AGIVE_GIVE | Set Every Of | EX_LIST (identifier-rounded) |
| EX_ARITH_ADD_CORR_TO | Add Corresponding Format | CORRESPONDING \$1{\$2= TO \$2} |
| \$1: ADD_CORR_WHAT | Add What | identifier |
| \$2: ADD_CORR_TOWHAT | To What | identifier-rounded |
| EX_ON_SIZE_ERROR | On Size Error Stmt | ON SIZE ERROR{\$1?\n\t\$1\b} |
| \$1: EX_ON_SIZE_ERROR__STMTS | Statements | STMT_LIST (statement) |
| EX_NOT_ON_SIZE_ERROR | Not On Size Error Stmt | NOT ON SIZE ERROR{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_SIZE_ERROR__STMTS | Statements | STMT_LIST (statement) |
| EX_END_ADD | End-Add | END-ADD |

10.6.5 ALTER statement

| | | |
|------------------------|-----------------|-----------------------------|
| STMT_ALTER | Alter Statement | ALTER \$1 |
| \$1: STMT_ALTER__ITEMS | Item List | EX_LIST (EX.ALTER_ITEM) |
| EX_ALTER_ITEM | Alter Item | \$1{\$2= TO PROCEED TO \$2} |
| \$1: ALTER_ITEM_FROM | Alter From | NAMED_OBJ_USE |
| \$2: ALTER_ITEM_TO | Alter To | NAMED_OBJ_USE |

10.6.6 CALL statement

| | | |
|----------------------|------------------------|--|
| STMT_CALL | Call Statement | CALL\$1{\$2= \$2}{\$3? \$3}{\$4? \$4}{\$5? \$5} \$ |
| \$1: CALL_MNEMONIC | Call Convention | NAMED_OBJ_USE EX_CALL_PROGRAM NULL |
| \$2: CALL_PROGRAM | Called Program | ident-liter |
| \$3: CALL_LINKAGE | Call Linkage (FSC) | EX_CALL_WITH_LINKAGE NULL |
| \$4: CALL_USING | Using Phrase | EX_CALL_USING NULL |
| \$5: CALL_GIVING | Giving Phrase (MF) | EX_CALL_GIVING NULL |
| \$6: CALL_ON_EXC | On Exception stmts | EX_ON_CALL_EXCEPTION NULL |
| \$7: CALL_NOT_ON_EXC | Not On Exception stmts | EX_NOT_ON_CALL_EXCEPTION NULL |
| \$8: CALL_END_CALL | End-Call | EX_END_CALL NULL |

| | | |
|-----------------|----------------|---------|
| EX_CALL_PROGRAM | Program phrase | PROGRAM |
|-----------------|----------------|---------|

| | | |
|--------------------------|---------------------|--|
| EX_CALL_USING | Caller Using Clause | USING{\$1= \$1} |
| \$1: EX_CALL_USING__ARGS | Arg List | EX_CALL_PARAM_LIST (EX_CALL_PARAM) NULL |

| | | |
|--------------------|---------------|-----|
| EX_CALL_PARAM_LIST | Call Arg List | @\n |
|--------------------|---------------|-----|

| | | |
|--------------------------|----------------------|---|
| EX_CALL_PARAM | Call Argument | {\$1?\$1 }{\$2=\$2}{\$3? \$3} |
| \$1: CALLARG_TYPE_PHRASE | Call Arg Type Phrase | EX_PT_BY_REFERENCE EX_PT_BY_CONTENT EX_PT_BY_VALUE NULL |
| \$2: CALLARG_IDENT_LITER | Arg Ident-Liter | ident-liter |
| \$3: CALLARG_SIZE | Size Phrase (MF) | EX_CALL_PARAM_SIZE NULL |
| \$4: CALLARG_TYPE | Arg Type | EX_PT_BY_REFERENCE EX_PT_BY_CONTENT EX_PT_BY_VALUE |

| | | |
|--------------------|-------------------------|--------------|
| EX_PT_BY_REFERENCE | By Reference Param Type | BY REFERENCE |
|--------------------|-------------------------|--------------|

| | | |
|------------------|-----------------------|------------|
| EX_PT_BY_CONTENT | By Content Param Type | BY CONTENT |
|------------------|-----------------------|------------|

| | | |
|----------------|---------------------|----------|
| EX_PT_BY_VALUE | By Value Param Type | BY VALUE |
|----------------|---------------------|----------|

| | | |
|-----------------------------------|---------------------|----------|
| EX_CALL_PARAM_SIZE | Call Parameter Size | SIZE \$1 |
| \$1: EX_CALL_PARAM_SIZE__VALUE | Value | literal |

| | | |
|---------------------------|---------------|--|
| EX_CALL_GIVING | Giving Clause | RETURNING \$1 |
| \$1: EX_CALL_GIVING__WHAT | Giving What | EX_CALL_GIVING_INT0 EX_CALL_GIVING_ADDRESS_OF |

| | | |
|------------------------------------|--------------------|------------|
| EX_CALL_GIVING_INT0 | Giving Into Phrase | \$1 |
| \$1: EX_CALL_GIVING_INT0__IDENT | Identifier | identifier |

| | | |
|--|--------------------------|----------------|
| EX_CALL_GIVING_ADDRESS_OF | Giving Address Of Phrase | ADDRESS OF \$1 |
| \$1: EX_CALL_GIVING_ADDRESS_OF__IDENT | Identifier | identifier |

| | | |
|-------------------------------------|------------------------|-----------------------------|
| EX_ON_CALL_EXCEPTION | On Call Exception Stmt | ON EXCEPTION{\$1?\n\t\$1\b} |
| \$1: EX_ON_CALL_EXCEPTION__STMTS | Statements | STMT_LIST (statement) |

| | | |
|---|----------------------------|---------------------------------|
| EX_NOT_ON_CALL_EXCEPTION | Not On Call Exception Stmt | NOT ON EXCEPTION{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_CALL_EXCEPTION__STMTS | Statements | STMT_LIST (statement) |

| | | |
|-------------|----------|----------|
| EX_END_CALL | End-Call | END-CALL |
|-------------|----------|----------|

10.6.7 CANCEL statement

| | | |
|------------------|------------------|-----------------------|
| STMT_CANCEL | Cancel Statement | CANCEL \$1 |
| \$1: CANCEL_PGMS | Cancel Programs | EX_LIST (ident-liter) |

10.6.8 CHAIN statement

| | | |
|----------------------|-----------------|--------------------------------|
| STMT_CHAIN | Chain Statement | CHAIN \$1{\$2? \$2}{\$3?\n\$3} |
| \$1: CHAIN_PROGRAM | Called Program | ident-liter |
| \$2: CHAIN_USING | Using Phrase | EX_CALL_USING NULL |
| \$3: CHAIN_END_CHAIN | End-Chain | EX_END_CHAIN NULL |

| | | |
|--------------|-----------|-----------|
| EX_END_CHAIN | End-Chain | END-CHAIN |
|--------------|-----------|-----------|

10.6.9 CLOSE statement

| | | |
|------------------|-----------------|-------------------------|
| STMT_CLOSE | Close Statement | CLOSE \$1 |
| \$1: CLOSE_ITEMS | Close Items | EX_LIST (EX_CLOSE_ITEM) |

| | | |
|------------------------|-------------|--|
| EX_CLOSE_ITEM | Close Item | \$1{\$2? \$2}{\$3? \$3} |
| \$1: CLOSE_ITEM_FILE | File | NAMED_OBJ_USE |
| \$2: CLOSE_ITEM_DEVICE | Device | IOCX_REEL IOCX_UNIT NULL |
| \$3: CLOSE_ITEM_DISP | Disposition | IOCX_WITH_LOCK IOCX_WITH_NO_REWIND IOCX_FOR_REMOVAL IOCX_WITH_DISP NULL |

| | | |
|------------------|-------------|-------------|
| IOCX_FOR_REMOVAL | For Removal | FOR REMOVAL |
|------------------|-------------|-------------|

| | | |
|----------------|-----------|-----------|
| IOCX_WITH_DISP | With Disp | WITH DISP |
|----------------|-----------|-----------|

| | | |
|----------------|-----------|-----------|
| IOCX_WITH_LOCK | With Lock | WITH LOCK |
|----------------|-----------|-----------|

| | | |
|---------------------|----------------|----------------|
| IOCX_WITH_NO_REWIND | With No Rewind | WITH NO REWIND |
|---------------------|----------------|----------------|

10.6.10 COMMIT statement

| | | |
|-------------|------------------|--------|
| STMT_COMMIT | Commit Statement | COMMIT |
|-------------|------------------|--------|

10.6.11 COMPUTE statement

| | | |
|--------------------------|-------------------|--|
| STMT_COMPUTE | Compute Statement | COMPUTE \$1{\$2 = \$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: COMPUTE.TO | Compute To | EX_LIST (identifier-rounded) |
| \$2: COMPUTE.FROM | Compute From | arith-expr |
| \$3: COMPUTE.ONSIZE | On Size Error | EX_ON_SIZE_ERROR NULL |
| \$4: COMPUTE.NOTONSIZE | Not On Size Error | EX_NOT_ON_SIZE_ERROR NULL |
| \$5: COMPUTE.END_COMPUTE | End-Compute | EX_END_COMPUTE NULL |

| | | |
|----------------|-------------|-------------|
| EX_END_COMPUTE | End-Compute | END-COMPUTE |
|----------------|-------------|-------------|

10.6.12 CONTINUE statement

| | | |
|---------------|----------|----------|
| STMT_CONTINUE | Continue | CONTINUE |
|---------------|----------|----------|

10.6.13 DELETE statement

| | | |
|-------------------------|-------------------------|--|
| STMT_DELETE | Delete Record Statement | DELETE \$1 RECORD{\$2? \$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: DELETE.RECORD | Record Name | NAMED_OBJ_USE |
| \$2: DELETE.TYPE | Type (DML) | EX_DML_ALL EX_DML_SELECTIVE EX_DML_ONLY NULL |
| \$3: DELETE.ONINVKEY | On Invalid Key | EX_ON_INVALID_KEY NULL |
| \$4: DELETE.NOTONINVKEY | Not On Invalid Key | EX_NOT_ON_INVALID_KEY NULL |
| \$5: DELETE.END_DELETE | End-Delete | EX_END_DELETE NULL |

| | | |
|---------------------|----------------|----------------------------|
| EX_ON_INVALID_KEY | On Invalid Key | INVALID KEY{\$1?\n\t\$1\b} |
| \$1: ONINVKEY_STMTS | Statements | STMT_LIST (statement) |

| | | |
|--------------------------------------|--------------------|--------------------------------|
| EX_NOT_ON_INVALID_KEY | Not On Invalid Key | NOT INVALID KEY{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_INVALID_KEY__STMTS | Statements | STMT_LIST (statement) |

| | | |
|---------------|------------|------------|
| EX_END_DELETE | End-Delete | END-DELETE |
|---------------|------------|------------|

10.6.14 DELETE FILE statement

| | | |
|---|-----------------------|-----------------|
| STMT_DELETE_FILE | Delete File Statement | DELETE FILE \$1 |
| \$1: STMT_DELETE_FILE__DEL_FILE_NAME | File Name | NAMED_OBJ_USE |

10.6.15 DISABLE statement

| | | |
|-------------------|-------------------|---|
| STMT_DISABLE | Disable Statement | DISABLE \$1{\$2= \$2}{\$3= WITH KEY \$3} |
| \$1: DISABLE_TYPE | Type | EX_ABLE_INPUT EX_ABLE_INPUT_TERMINAL EX_ABLE_I_O_TERMINAL EX_ABLE_OUTPUT |
| \$2: DISABLE_CD | CD name | NAMED_OBJ_USE |
| \$3: DISABLE_KEY | Key | ident-liter |

| | | |
|---------------|-------|-------|
| EX_ABLE_INPUT | Input | INPUT |
|---------------|-------|-------|

| | | |
|------------------------|----------------|----------------|
| EX_ABLE_INPUT_TERMINAL | Input Terminal | INPUT TERMINAL |
|------------------------|----------------|----------------|

| | | |
|----------------------|--------------|--------------|
| EX_ABLE_I_O_TERMINAL | I-O Terminal | I-O TERMINAL |
|----------------------|--------------|--------------|

| | | |
|----------------|--------|--------|
| EX_ABLE_OUTPUT | Output | OUTPUT |
|----------------|--------|--------|

10.6.16 DISPLAY statement

| | | |
|-----------------------|-------------------|--|
| STMT_DISPLAY | Display Statement | DISPLAY \$1 |
| \$1: DISPLAY_ITEMS | Display Item List | DISPLAY_ITEM_LIST (DISPLAY_FUNC_ITEM DISPLAY_ANSI_ITEM DISPLAY_SCREEN_ITEM) |
| \$2: DISPLAY_ONEXC | On Exception | ON_DISPLAY_EXCEPTION NULL |
| \$3: DISPLAY_NOTONEXC | Not On Exception | NOT_ON_DISPLAY_EXCEPTION NULL |
| \$4: DISPLAY_END | End-Display | EX_END_DISPLAY NULL |

| | | |
|-------------------|-------------------|-----|
| DISPLAY_ITEM_LIST | Display Item List | @\n |
|-------------------|-------------------|-----|

| | | |
|-------------------------------------|----------------------------|--|
| DISPLAY_FUNC_ITEM | Misc Function Display Item | { \$1? \$1 } { \$2? \$2 } { \$3? \$3 } |
| \$1: IX_DSPL_FUNC_DATA_NAME | Display Value | ident-liter |
| \$2: IX_DSPL_FUNC_UPON | Upon What | DSPL_UPON NULL |
| \$3: IX_DSPL_FUNC_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |

| | | |
|-------------------------------------|---------------------------|---|
| DISPLAY_ANSI_ITEM | ANSI Console Display Item | { \$1? \$1 } { \$2? \$2 } { \$3? \$3 } { \$4? \$4 } |
| \$1: IX_DSPL_ANSI_DATA_NAME | Display Value | ident-liter |
| \$2: IX_DSPL_ANSI_UPON | Upon What | DSPL_UPON NULL |
| \$3: IX_DSPL_ANSI_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$4: IX_DSPL_ANSI_NO_ADVANCING | Advancing | DSPL_NO_ADVANCING NULL |

| | | |
|-----------------------------------|---------------------|---|
| DISPLAY_SCREEN_ITEM | Screen Display Item | { \$1? \$1 } { \$2? \$2 } { \$3? \$3 } { \$4? \$4 } { \$5? |
| \$1: IX_DSPL_POSN_SPEC | Position Spec (MS) | EX_ADIS_MS_POSN NULL |
| \$2: IX_DSPL_DATA_NAME | Value To Display | ident-liter DSPL_MS_ERASE |
| \$3: IX_DSPL_LINE_NUMBER | Line Number | DSPL_LINE_NUMBER NULL |
| \$4: IX_DSPL_COLUMN_NUMBER | Column Number | DSPL_COLUMN_NUMBER NULL |
| \$5: IX_DSPL_AT_POS | At | DSPL_AT DSPL_POSITION NULL |
| \$6: IX_DSPL_FROM_UPON | From or Upon | DSPL_UPON DSPL_UNIT NULL |
| \$7: IX_DSPL_MODE_IS_BLOCK | Mode Is Block | DSPL_MODE_IS_BLOCK NULL |
| \$8: IX_DSPL_WITH | With Word | DSPL_WITH NULL |
| \$9: IX_DSPL_BELL | Bell | DSPL_BELL NULL |
| \$10: IX_DSPL_BLINK | Blink | DSPL_BLINK NULL |
| \$11: IX_DSPL_ERASE | Erase | DSPL_ERASE NULL |
| \$12: IX_DSPL_GRID | Grid | DSPL_GRID NULL |
| \$13: IX_DSPL_HIGH_LOW | Highlight/Lowlight | DSPL_HIGHLIGHT DSPL_HIGH DSPL_LOWLIGHT DSPL_LOW NULL |
| \$14: IX_DSPL_LEFTLINE | Leftline | DSPL_LOWLIGHT NULL |
| \$15: IX_DSPL_OVERLINE | Overline | DSPL_OVERLINE NULL |
| \$16: IX_DSPL_REVERSE | Reverse-Video | DSPL_REVERSE_VIDEO DSPL_REVERSE NULL |
| \$17: IX_DSPL_SIZE | Size | DSPL_SIZE NULL |
| \$18: IX_DSPL_UNDERLINE | Underline | DSPL_UNDERLINE NULL |
| \$19: IX_DSPL_FOREGROUND_COLOR | Foreground-Color | DSPL_FOREGROUND_COLOR NULL |
| \$20: IX_DSPL_BACKGROUND_COLOR | Background-Color | DSPL_FOREGROUND_COLOR NULL |
| \$21: IX_DSPL_CONTROL | Control | DSPL_CONTROL NULL |

| | | |
|---------------------------------|---------------------------|--|
| DISPLAY_SCREEN_ITEM | (continued) | (continued) |
| \$22: IX_DSPL_PROMPT | Prompt | SCRD_PROMPT ACCP_PROMPT NULL |
| \$23: IX_DSPL_AUTO | Auto | ACCP_AUTO NULL |
| \$24: IX_DSPL_FULL | Full | ACCP_FULL NULL |
| \$25: IX_DSPL_REQUIRED | Required | ACCP_REQUIRED NULL |
| \$26: IX_DSPL_SECURE | Secure | ACCP_SECURE NULL |
| \$27: IX_DSPL_ZERO_FILL | Zero-Fill | ACCP_ZERO_FILL NULL |
| \$28: IX_DSPL_JUST | Just | SCRD_JUSTIFIED ACCP_LEFT_JUSTIFY ACCP_RIGHT_JUSTIFY NULL |
| \$29: IX_DSPL_SIGN | Sign | SCRD_SIGN ACCP_TRAILING_SIGN NULL |
| \$30: IX_DSPL_BLANK | Blank | DSPL_BLANK NULL |
| \$31: IX_DSPL_UNINLINED_COPY | CopyBegin/End/SyntaxError | copy-expr NULL |
| \$32: IX_DSPL_NO_ADVANCING | Advancing Phrase (ANSI) | DSPL_NO_ADVANCING NULL |

| | | |
|---------------|-------|-------|
| DSPL_MS_ERASE | Erase | ERASE |
|---------------|-------|-------|

| | | |
|---------------------------------|--------------------|-------------|
| DSPL_LINE_NUMBER | Line Number Clause | LINE \$1 |
| \$1: DSPL_LINE_NUMBER__VALUE | Line Number | ident-liter |

| | | |
|-----------------------------------|----------------------|-------------|
| DSPL_COLUMN_NUMBER | Column Number Clause | COLUMN \$1 |
| \$1: DSPL_COLUMN_NUMBER__VALUE | Column Number | ident-liter |

| | | |
|---------------------------|-----------------|--------------|
| DSPL_POSITION | Position Clause | POSITION \$1 |
| \$1: DSPL_POSITION__VALUE | Position | ident-liter |

| | | |
|---------------------|-------------|-------------|
| DSPL_AT | At Clause | AT \$1 |
| \$1: DSPL_AT__VALUE | At Position | ident-liter |

| | | |
|---------------------|-------------|-------------|
| DSPL_UPON | Upon Clause | UPON \$1 |
| \$1: DSPL_UPON__IOF | IO function | io-function |

| | | |
|--------------------|---------------|---------------|
| DSPL_MODE_IS_BLOCK | Mode Is Block | MODE IS BLOCK |
|--------------------|---------------|---------------|

| | | |
|-----------|------|------|
| DSPL_WITH | With | WITH |
|-----------|------|------|

| | | |
|-----------|------|------|
| DSPL_HIGH | High | HIGH |
|-----------|------|------|

| | | |
|----------|-----|-----|
| DSPL_LOW | Low | LOW |
|----------|-----|-----|

| | | |
|--------------|---------|---------|
| DSPL_REVERSE | Reverse | REVERSE |
|--------------|---------|---------|

| | | |
|-----------------------|------|-------------|
| DSPL_UNIT | Unit | UNIT \$1 |
| \$1: DSPL_UNIT__VALUE | Unit | ident-liter |

| | | |
|-------------------|--------------|--------------|
| DSPL_NO_ADVANCING | No Advancing | NO ADVANCING |
|-------------------|--------------|--------------|

| | | |
|----------------------|----------------------|-----------------------------|
| ON_DISPLAY_EXCEPTION | On Display Exception | ON EXCEPTION{\$1?\n\t\$1\b} |
| \$1: ONDSPLXC_STMTS | Statements | STMT_LIST (statement) |

| | | |
|--|--------------------------|---------------------------------|
| NOT_ON_DISPLAY_EXCEPTION | Not On Display Exception | NOT ON EXCEPTION{\$1?\n\t\$1\b} |
| \$1: NOT_ON_DISPLAY_EXCEPTION_STMTS | Statements | STMT_LIST (statement) |

| | | |
|----------------|-------------|-------------|
| EX_END_DISPLAY | End-DISPLAY | END-DISPLAY |
|----------------|-------------|-------------|

10.6.17 DIVIDE statement

| | | |
|------------------------|-------------------|---|
| STMT_DIVIDE | Divide Statement | DIVIDE \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: DIVIDE_FORMAT | Divide Format | EX_ARITH_DIV_INT0 EX_ARITH_DIV_INT0_GIVING EX_ARITH_DIV_BY_GIVING |
| \$2: DIVIDE_ONSIZE | On Size Error | EX_ON_SIZE_ERROR NULL |
| \$3: DIVIDE_NOTONSIZE | Not On Size Error | EX_NOT_ON_SIZE_ERROR NULL |
| \$4: DIVIDE_END_DIVIDE | End-Divide | EX_END_DIVIDE NULL |

| | | |
|--------------------|--------------------|------------------------------|
| EX_ARITH_DIV_INT0 | Divide Into Format | \$1{\$2= INT0 \$2} |
| \$1: DIV_INT0_WHAT | What | ident-liter |
| \$2: DIV_INT0_INT0 | Into What | EX_LIST (identifier-rounded) |

| | | |
|--------------------------|---------------------------|---|
| EX_ARITH_DIV_INT0_GIVING | Divide Into Giving Format | \$1{\$2= INT0 \$2}{\$3= GIVING \$3}{\$4? \$4} |
| \$1: DIV_INT0GV_WHAT | What | ident-liter |
| \$2: DIV_INT0GV_INT0 | Into | ident-liter |
| \$3: DIV_INT0GV_GIVING | Giving | EX_LIST (identifier-rounded) |
| \$4: DIV_INT0GV_REM | Remainder | EX_ARITH_DIV_REMAINDER NULL |

| | | |
|------------------------|-------------------------|---|
| EX_ARITH_DIV_BY_GIVING | Divide By Giving Format | \$1{\$2= BY \$2}{\$3= GIVING \$3}{\$4? \$4} |
| \$1: DIV_BYGV_WHAT | What | ident-liter |
| \$2: DIV_BYGV_BY | By | ident-liter |
| \$3: DIV_BYGV_GIVING | Giving | EX_LIST (identifier-rounded) |
| \$4: DIV_BYGV_REM | Remainder | EX_ARITH_DIV_REMAINDER NULL |

| | | |
|---------------------------------------|------------------|---------------|
| EX_ARITH_DIV_REMAINDER | Remainder Clause | REMAINDER \$1 |
| \$1: EX_ARITH_DIV_REMAINDER__IDENT | Identifier | identifier |

| | | |
|---------------|------------|------------|
| EX_END_DIVIDE | End-Divide | END-DIVIDE |
|---------------|------------|------------|

10.6.18 ENABLE statement

| | | |
|------------------|------------------|---|
| STMT_ENABLE | Enable Statement | ENABLE \$1{\$2= \$2}{\$3= WITH KEY \$3} |
| \$1: ENABLE_TYPE | Type | EX_ABLE_INPUT EX_ABLE_INPUT_TERMINAL EX_ABLE_I_O_TERMINAL EX_ABLE_OUTPUT |
| \$2: ENABLE_CD | CD name | NAMED_OBJ_USE |
| \$3: ENABLE_KEY | Key | ident-liter |

10.6.19 ENTRY statement

| | | |
|---------------------------|-------------------------------|--|
| STMT_ENTRY | Entry Statement | \bENTRY \$1{\$2? \$2}{\$3? \$3}{\$4? \$4}{\$5? \$5} |
| \$1: STMT_ENTRY__NAME | Entry Name Defined | STR_CONST |
| \$2: STMT_ENTRY__MNEMONIC | Call Convention Mnemonic (MF) | NAMED_OBJ_USE NULL |
| \$3: STMT_ENTRY__LINKAGE | Call Linkage (FSC) | EX_CALL_WITH_LINKAGE NULL |
| \$4: STMT_ENTRY__USING | Using Clause | EX_PROC_USING EX_PROC_CHAINING NULL |
| \$5: STMT_ENTRY__GIVING | Giving Clause | EX_CALL_GIVING NULL |

10.6.20 EVALUATE statement

| | | |
|-----------------------|--------------------|---|
| STMT_EVALUATE | Evaluate Statement | EVALUATE \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: EVAL_SUBJECTS | Subjects | EX_EVAL_SUBJ_LIST (eval-condition) |
| \$2: EVAL_WHEN_BLOCKS | When Blocks | EX_EVAL_WHEN_BLOCK_LIST (EX_EVAL_WHEN_BLOCK) |
| \$3: EVAL_WHEN_OTHER | When Other | EX_EVAL_WHEN_OTHER NULL |
| \$4: EVAL_END_EVAL | End-Evaluate | EX_END_EVALUATE NULL |

| | | |
|-------------------|-----------------------|--------|
| EX_EVAL_SUBJ_LIST | Evaluate Subject List | @%ALSO |
|-------------------|-----------------------|--------|

| | | |
|------------------|--------------------|--|
| • eval-condition | Evaluate Condition | condition EX_ANY EX_TRUE EX_FALSE |
|------------------|--------------------|--|

| | | |
|-------------------------|--------------------------|-------|
| EX_EVAL_WHEN_BLOCK_LIST | Evaluate When Block List | \r@\n |
|-------------------------|--------------------------|-------|

| | | |
|--------------------|---------------------|---|
| EX_EVAL_WHEN_BLOCK | Evaluate When Block | \r\$1{\$2?\n\t\$2\b} |
| \$1: EVAL_WB_WPS | Phrase List | EX_EVAL_WHEN_PHRASE_LIST (EX_EVAL_WHEN_PHRASE) |
| \$2: EVAL_WB_STMTS | Statements | STMT_LIST (statement) |

| | | |
|--------------------------|---------------------------|-------|
| EX_EVAL_WHEN_PHRASE_LIST | Evaluate When Phrase List | \r@\n |
|--------------------------|---------------------------|-------|

| | | |
|----------------------------------|----------------------|----------------------------------|
| EX_EVAL_WHEN_PHRASE | Evaluate When Phrase | WHEN \$1 |
| \$1: EX_EVAL_WHEN_PHRASE__OBS | Object List | EX_EVAL_OBJ_LIST (EX_EXPR_RANGE) |

| | | |
|------------------|----------------------|--------|
| EX_EVAL_OBJ_LIST | Evaluate Object List | @%ALSO |
|------------------|----------------------|--------|

| | | |
|----------------------|------------------|--------------------------------|
| EX_EXPR_RANGE | Expression Range | \$1{\$2?%THRU \$2} |
| \$1: EXPR_RANGE_FROM | From | eval-condition arith-expr |
| \$2: EXPR_RANGE_TO | To | arith-expr |

| | | |
|---|-------------------|---------------------------|
| EX_EVAL_WHEN_OTHER | When Other Clause | WHEN OTHER{\$1?\n\t\$1\b} |
| \$1: EX_EVAL_WHEN_OTHER__EVAL_WO_STMTS | Statements | STMT_LIST (statement) |

| | | |
|-----------------|--------------|--------------|
| EX_END_EVALUATE | End-Evaluate | END-EVALUATE |
|-----------------|--------------|--------------|

10.6.21 EXAMINE statement

| | | |
|---------------------------|----------------------------|--|
| STMT_EXAMINE_TALLYING | Examine Tallying Statement | EXAMINE \$1{\$2= TALLYING \$2}{\$3? \$3} |
| \$1: EXAMINE_TALLY_DATA | Data Item | identifier |
| \$2: EXAMINE_TALLY_WHAT | Examine Option | EX_STR_EXAM_PHRASE NULL |
| \$3: EXAMINE_TALLY_REPLBY | Replacing By | EX_STR_REPLACING_BY NULL |

| | | |
|------------------------|-----------------------------|---|
| STMT_EXAMINE_REPLACING | Examine Replacing Statement | EXAMINE \$1{\$2= REPLACING \$2}{\$3= BY |
| \$1: EXAMINE_REPL_DATA | Data Item | identifier |
| \$2: EXAMINE_REPL_FROM | From | EX_STR_EXAM_PHRASE NULL |
| \$3: EXAMINE_REPL_TO | To | literal |

| | | |
|------------------------------------|---------------------|------------------|
| EX_STR_REPLACING_BY | Replacing By Clause | REPLACING BY \$1 |
| \$1: EX_STR_REPLACING_BY__VALUE | Value | literal |

| | | |
|-------------------------|----------------|--|
| EX_STR_EXAM_PHRASE | Examine Phrase | \$1{\$2= \$2} |
| \$1: EXAMINE_PHR_OPTION | Examine Option | EX_STR_UNTIL_FIRST EX_STR_ALL EX_STR_LEADING EX_STR_FIRST |
| \$2: EXAMINE_PHR_VALUE | Value | literal |

| | | |
|--------------------|-------------|-------------|
| EX_STR_UNTIL_FIRST | Until First | UNTIL FIRST |
|--------------------|-------------|-------------|

10.6.22 EXEC statement

| | | |
|------------------|----------------|--|
| STMT_EXEC_IN_DCL | Exec Statement | EXEC \$1\n\t{\$2=\$2}\b\n[END-EXEC.] |
| \$1: EXEC_SUBSYS | Subsystem | CHAR_STRING |
| \$2: EXEC_LINES | Lines | EX_EXEC_LINE_LIST (EX_EXEC_TOKEN_LIST (CHAR_STRING)) |

| | | |
|------------------------|----------------|--|
| STMT_EXEC | Exec Statement | EXEC \$1\n\t{\$2=\$2}\b\n[END-EXEC] |
| \$1: STMT_EXEC__SUBSYS | Subsystem | CHAR_STRING |
| \$2: STMT_EXEC__LINES | Lines | EX_EXEC_LINE_LIST (EX_EXEC_TOKEN_LIST (CHAR_STRING)) |

| | | |
|-------------------|----------------|-----|
| EX_EXEC_LINE_LIST | Exec Line List | @\n |
|-------------------|----------------|-----|

| | | |
|--------------------|-----------------|----|
| EX_EXEC_TOKEN_LIST | Exec Token List | @! |
|--------------------|-----------------|----|

10.6.23 EXHIBIT statement

| | | |
|-------------------|-------------------|---------------------------------------|
| STMT_EXHIBIT | Exhibit Statement | EXHIBIT{\$1? \$1}{\$2? \$2}{\$3? \$3} |
| \$1: EXHB_CHANGED | Changed | EX_EXHIBIT_CHANGED NULL |
| \$2: EXHB_NAMED | Named | EX_EXHIBIT_NAMED NULL |
| \$3: EXHB_VALUES | Values | EX_LIST (ident-liter) |

| | | |
|--------------------|----------------|---------|
| EX_EXHIBIT_CHANGED | Changed Clause | CHANGED |
|--------------------|----------------|---------|

| | | |
|------------------|--------------|-------|
| EX_EXHIBIT_NAMED | Named Clause | NAMED |
|------------------|--------------|-------|

10.6.24 EXIT statement

| | | |
|-----------|------------|------|
| STMT_EXIT | Exit No-Op | EXIT |
|-----------|------------|------|

10.6.25 EXIT PROGRAM statement

| | | |
|-----------------------------------|------------------------|------------------------|
| STMT_EXIT_PROGRAM | Exit Program Statement | EXIT PROGRAM{\$1? \$1} |
| \$1: STMT_EXIT_PROGRAM__GIVING | Giving | EX_EXIT_GIVING NULL |

| | | |
|---------------------------|---------------|--------------------|
| EX_EXIT_GIVING | Giving Clause | GIVING \$1 |
| \$1: EX_EXIT_GIVING__WHAT | Giving What | EX_EXIT_ARG NULL |

| | | |
|------------------------|---------------|---------------------------|
| EX_EXIT_ARG | Exit Argument | \$1{\$2? \$2} |
| \$1: EX_EXIT_ARG_VALUE | Value | ident-liter NULL |
| \$2: EX_EXIT_ARG_SIZE | Size | EX_CALL_PARAM_SIZE NULL |

10.6.26 EXIT PERFORM statement

| | | |
|----------------------------------|------------------------|------------------------|
| STMT_EXIT_PERFORM | Exit Perform Statement | EXIT PERFORM{\$1? \$1} |
| \$1: STMT_EXIT_PERFORM__CYCLE | Cycle | EX_EXIT_CYCLE NULL |

| | | |
|---------------|-------|-------|
| EX_EXIT_CYCLE | Cycle | CYCLE |
|---------------|-------|-------|

10.6.27 EXIT PARAGRAPH statement

| | | |
|---------------------|--------------------------|----------------|
| STMT_EXIT_PARAGRAPH | Exit Paragraph Statement | EXIT PARAGRAPH |
|---------------------|--------------------------|----------------|

| | | |
|-------------------|------------------------|--------------|
| STMT_EXIT_SECTION | Exit Section Statement | EXIT SECTION |
|-------------------|------------------------|--------------|

10.6.28 GENERATE statement

| | | |
|----------------------------|--------------------|---------------|
| STMT_GENERATE | Generate Statement | GENERATE \$1 |
| \$1: STMT_GENERATE__REPORT | Report | NAMED_OBJ_USE |

10.6.29 GOBACK statement

| | | |
|--------------------------|------------------|-----------------------|
| STMT_GOBACK | Goback Statement | GOBACK{\$1? \$1} |
| \$1: STMT_GOBACK__GIVING | Giving | EX_EXIT_GIVING NULL |

10.6.30 GOTO statement

| | | |
|-----------------------|-----------------|---------------|
| STMT_GOTO | Go To Statement | GO TO \$1 |
| \$1: STMT_GOTO__LABEL | Where To Label | NAMED_OBJ_USE |

| | | |
|---------------------|---------------------------|-------------------------|
| STMT_GOTO_DEPENDING | Go To Depending Statement | GO TO \$1{\$2= \$2} |
| \$1: GOTODEP_LABELS | Where To Label List | EX_LIST (NAMED_OBJ_USE) |
| \$2: GOTODEP_DEPON | Depending On Clause | EX_DEPENDING_ON |

10.6.31 IF statement

| | | |
|-------------------|--------------|---|
| STMT_IF | If Statement | IF \$1{\$2?\n\t\$2\b}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: IF_CONDITION | Condition | condition |
| \$2: IF_THEN | Then Branch | STMT_LIST (statement) |
| \$3: IF_ELSE | Else Branch | EX_ELSE NULL |
| \$4: IF_END_IF | End-If | EX_END_IF NULL |

| | | |
|---------------------|-------------|-----------------------|
| EX_ELSE | Else Branch | ELSE{\$1?\n\t\$1\b} |
| \$1: EX_ELSE__STMTS | Statements | STMT_LIST (statement) |

| | | |
|-----------|--------|--------|
| EX_END_IF | End-If | END-IF |
|-----------|--------|--------|

10.6.32 INITIALIZE statement

| | | |
|------------------------|----------------------|--------------------------|
| STMT_INITIALIZE | Initialize Statement | INITIALIZE \$1{\$2? \$2} |
| \$1: INITIALIZE_WHAT | What | IDENT_LIST (identifier) |
| \$2: INITIALIZE_REPLNG | Replacing Phrase | EX_INIT_REPLACING NULL |

| | | |
|------------------------------------|-----------------------|---------------------------------------|
| EX_INIT_REPLACING | Init Replacing Clause | REPLACING \$1 |
| \$1: EX_INIT_REPLACING__PHRASES | Replacing Phrase List | EX_LIST (EX_STR_REPLACING_DATA_BY) |

| | | |
|--------------------------|------------------|---|
| EX_STR_REPLACING_DATA_BY | Replacing Phrase | \$1{\$2= DATA BY \$2} |
| \$1: INIT_REPLNG_CAT | Category | EX_STR_ALPHABETIC EX_STR_ALPHANUMERIC EX_STR_NUMERIC EX_STR_ALPHANUMERIC_EDITED EX_STR_NUMERIC_EDITED |
| \$2: INIT_REPLNG_BYWHAT | By What | ident-liter |

| | | |
|-------------------|------------|------------|
| EX_STR_ALPHABETIC | Alphabetic | ALPHABETIC |
|-------------------|------------|------------|

| | | |
|---------------------|--------------|--------------|
| EX_STR_ALPHANUMERIC | Alphanumeric | ALPHANUMERIC |
|---------------------|--------------|--------------|

| | | |
|----------------------------|---------------------|---------------------|
| EX_STR_ALPHANUMERIC_EDITED | Alphanumeric-Edited | ALPHANUMERIC-EDITED |
|----------------------------|---------------------|---------------------|

| | | |
|----------------|---------|---------|
| EX_STR_NUMERIC | Numeric | NUMERIC |
|----------------|---------|---------|

| | | |
|-----------------------|----------------|----------------|
| EX_STR_NUMERIC_EDITED | Numeric-Edited | NUMERIC-EDITED |
|-----------------------|----------------|----------------|

10.6.33 INITIATE statement

| | | |
|--------------------------------|--------------------|-------------------------|
| STMT_INITIATE | Initiate Statement | INITIATE \$1 |
| \$1: STMT_INITIATE__REPORTS | Reports | EX_LIST (NAMED_OBJ_USE) |

10.6.34 INSPECT statement

| | | |
|------------------------|-------------------|---------------------------------|
| STMT_INSPECT | Inspect Statement | INSPECT \$1{\$2? \$2}{\$3? \$3} |
| \$1: INSPECT_WHAT | Identifier | identifier |
| \$2: INSPECT_TALLYING | Tallying | EX_STR_TALLYING NULL |
| \$3: INSPECT_REPLACING | Replacing | EX_INSPECT_REPLACING NULL |

| | | |
|--|----------------------|---|
| EX_STR_TALLYING | Tallying Clause | TALLYING \$1 |
| \$1: EX_STR_TALLYING__INSPEC_TALLYING_CLAUSES | Tallying Clause List | EX_STR_TALLY_CLAUSE_LIST (EX_STR_TALLY_CLAUSE) |

| | | |
|--------------------------|----------------------|---|
| EX_STR_TALLY_CLAUSE_LIST | Tallying Clause List | @ |
|--------------------------|----------------------|---|

| | | |
|---------------------------|-------------------|--|
| EX_STR_TALLY_CLAUSE | Tallying Clause | \$1{\$2= FOR \$2} |
| \$1: INSPEC_TALLY_IDENT | Identifier | identifier |
| \$2: INSPEC_TALLY_PHRASES | Tally Phrase List | EX_STR_TALLY_PHRASE_LIST (EX_STR_TALLY_PHRASE EX_STR_TALLY_UNIT) |

| | | |
|--------------------------|-------------------|---|
| EX_STR_TALLY_PHRASE_LIST | Tally Phrase List | @ |
|--------------------------|-------------------|---|

| | | |
|-------------------------------------|-----------------|---|
| EX_STR_TALLY_PHRASE | Tally Phrase | \$1{\$2= \$2} |
| \$1: INSPECT_TALLY_PHRASE_ALLEAD | All/Leading | EX_STR_ALL EX_STR_LEADING |
| \$2: INSPECT_TALLY_PHRASE_UNITS | Tally Unit List | EX_STR_TALLY_UNIT_LIST (EX_STR_TALLY_UNIT) |

| | | |
|------------------------|-----------------|---|
| EX_STR_TALLY_UNIT_LIST | Tally Unit List | @ |
|------------------------|-----------------|---|

| | | |
|-----------------------------------|---------------------------|---|
| EX_STR_TALLY_UNIT | Tally Unit | \$1{\$2? \$2} |
| \$1: INSPECT_TALLY_UNIT_VALUE | Value | ident-liter EX_STR_TALLY_CHARACTERS |
| \$2: INSPECT_TALLY_UNIT_BAINIT | Before/After Initial List | EX_STR_BA_INITIAL_LIST (EX_STR_BA_INITIAL) |

| | | |
|------------------------|---------------------------|---|
| EX_STR_BA_INITIAL_LIST | Before/After Initial List | @ |
|------------------------|---------------------------|---|

| | | |
|------------------------------------|-----------------------------|-------------------------|
| EX_STR_BA_INITIAL | Before/After Initial Phrase | \$1{\$2= INITIAL \$2} |
| \$1: INSPECT_TALLY_BAINIT_BA | Before/After | EX_BEFORE EX_AFTER |
| \$2: INSPECT_TALLY_BAINIT_VALUE | Value | ident-liter |

| | | |
|-----------|--------|--------|
| EX_BEFORE | Before | BEFORE |
|-----------|--------|--------|

| | | |
|----------|-------|-------|
| EX_AFTER | After | AFTER |
|----------|-------|-------|

| | | |
|-------------------------|------------|------------|
| EX_STR_TALLY_CHARACTERS | Characters | CHARACTERS |
|-------------------------|------------|------------|

| | | |
|---------------------------------------|--------------------------|--|
| EX_INSPECT_REPLACING | Inspect Replacing Clause | REPLACING \$1 |
| \$1: EX_INSPECT_REPLACING__PHRASES | Replacing Phrase List | EX_STR_REPLACE_PHRASE_LIST (EX_STR_REPLACE_UNIT EX_STR_REPLACE_PHRASE) |

| | | |
|----------------------------|-----------------------|---|
| EX_STR_REPLACE_PHRASE_LIST | Replacing Phrase List | @ |
|----------------------------|-----------------------|---|

| | | |
|---------------------------------------|---------------------|---|
| EX_STR_REPLACE_PHRASE | Replacing Phrase | \$1{\$2= \$2} |
| \$1: INSPECT_REPLACE_PHRASE_ALLEAD | All/Leading/First | EX_STR_ALL EX_STR_LEADING EX_STR_FIRST |
| \$2: INSPECT_REPLACE_PHRASE_UNITS | Replacing Unit List | EX_STR_REPLACE_UNIT_LIST (EX_STR_REPLACE_UNIT) |

| | | |
|--------------------------|---------------------|---|
| EX_STR_REPLACE_UNIT_LIST | Replacing Unit List | @ |
|--------------------------|---------------------|---|

| | | |
|-------------------------------------|---------------------------|---|
| EX_STR_REPLACE_UNIT | Replacing Unit | \$1{\$2= BY \$2}{\$3? \$3} |
| \$1: INSPECT_REPLACE_UNIT_FROM | What | ident-liter EX_STR_REPL_CHARACTERS |
| \$2: INSPECT_REPLACE_UNIT_TO | By What | ident-liter |
| \$3: INSPECT_REPLACE_UNIT_BAINIT | Before/After Initial List | EX_STR_BA_INITIAL_LIST (EX_STR_BA_INITIAL) |

| | | |
|------------------------|------------|------------|
| EX_STR_REPL_CHARACTERS | Characters | CHARACTERS |
|------------------------|------------|------------|

| | | |
|------------|-----|-----|
| EX_STR_ALL | All | ALL |
|------------|-----|-----|

| | | |
|----------------|---------|---------|
| EX_STR_LEADING | Leading | LEADING |
|----------------|---------|---------|

| | | |
|--------------|-------|-------|
| EX_STR_FIRST | First | FIRST |
|--------------|-------|-------|

| | | |
|--------------------------|---------------------------|---|
| STMT_INSPECT_CONVERTING | Inspect Statement | INSPECT \$1{\$2= CONVERTING \$2}{\$3= TO |
| \$1: INSPECT_CONV_TARGET | Target | identifier |
| \$2: INSPECT_CONV_FROM | From | ident-liter |
| \$3: INSPECT_CONV_TO | To | ident-liter |
| \$4: INSPECT_CONV_BAINIT | Before/After Initial List | EX_STR_BA_INITIAL_LIST (EX_STR_BA_INITIAL) |

10.6.35 MERGE statement

| | | |
|------------------------|---------------------------|---|
| STMT_MERGE | Merge Statement | MERGE \$1{\$2= \$2}{\$3? \$3}{\$4= \$4}{\$5= |
| \$1: MERGE_TARGET_FILE | Target File | NAMED_OBJ_USE |
| \$2: MERGE_SORT_KEYS | Sort Keys | EX_LIST (EX_SORT_KEY) NULL |
| \$3: MERGE_COLLSEQ | Collating Sequence | EX_SORT_COLL_SEQCE NULL |
| \$4: MERGE_USING | Source Files | EX_SORT_USING |
| \$5: MERGE_OUTGIVING | Output Procedure / Giving | EX_OUTPUT_PROC_IS EX_GIVING NULL |

10.6.36 MOVE statement

| | | |
|------------------|----------------|-------------------------|
| STMT_MOVE | Move Statement | MOVE \$1{\$2= TO \$2} |
| \$1: MOVE_SOURCE | Source | ident-liter |
| \$2: MOVE_DESTS | Destinations | IDENT_LIST (identifier) |

| | | |
|-----------------------|----------------|-------------------------------------|
| STMT_MOVE_CORR | Move Statement | MOVE CORRESPONDING \$1{\$2= TO \$2} |
| \$1: MOVE_CORR_SOURCE | Source | ident-liter |
| \$2: MOVE_CORR_DESTS | Destinations | IDENT_LIST (identifier) |

10.6.37 MULTIPLY statement

| | | |
|----------------------------|--------------------|--|
| STMT_MULTIPLY | Multiply Statement | MULTIPLY \$1{\$2?\n \$2}{\$3?\n \$3}{\$4?\n \$ |
| \$1: MULTIPLY_FORMAT | Multiply Format | EX_ARITH_MUL_BY EX_ARITH_MUL_BY_GIVING |
| \$2: MULTIPLY_ONSIZE | On Size Error | EX_ON_SIZE_ERROR NULL |
| \$3: MULTIPLY_NOTONSIZE | Not On Size Error | EX_NOT_ON_SIZE_ERROR NULL |
| \$4: MULTIPLY_END_MULTIPLY | End-Multiply | EX_END_MULTIPLY NULL |

| | | |
|-----------------------|--------------------------------|------------------------------|
| EX_ARITH_MUL_BY | Multiply By Format | \$1{\$2= BY \$2} |
| \$1: MULTIPLY_BY_ARG1 | First operand | ident-liter |
| \$2: MULTIPLY_BY_DEST | Second Operand And Destination | EX_LIST (identifier-rounded) |

| | | |
|---------------------------|---------------------------|-----------------------------------|
| EX_ARITH_MUL_BY_GIVING | Multiply By Giving Format | \$1{\$2= BY \$2}{\$3= GIVING \$3} |
| \$1: MULTIPLY_BYGIVE_ARG1 | First operand | ident-liter |
| \$2: MULTIPLY_BYGIVE_ARG2 | Second operand | ident-liter |
| \$3: MULTIPLY_BYGIVE_DEST | Destination | EX_LIST (identifier-rounded) |

| | | |
|-----------------|--------------|--------------|
| EX_END_MULTIPLY | End-Multiply | END-MULTIPLY |
|-----------------|--------------|--------------|

10.6.38 NEXT-SENTENCE statement

| | | |
|--------------------|----------------|---------------|
| STMT_NEXT_SENTENCE | Next Statement | NEXT SENTENCE |
|--------------------|----------------|---------------|

10.6.39 NOTE statement

| | | |
|-----------|-----------------------|--|
| STMT_NOTE | Note Statement (OSVS) | |
|-----------|-----------------------|--|

10.6.40 OPEN statement

| | | |
|------------------------|------------------|--------------------------|
| STMT_OPEN | Open Statement | OPEN \$1 |
| \$1: STMT_OPEN_CLAUSES | Open Clause List | EX_LIST (EX_OPEN_CLAUSE) |

| | | |
|-----------------------|---------------------|--|
| EX_OPEN_CLAUSE | Open Clause | { \$1? \$1 } { \$2 = \$2 } { \$3 = \$3 } |
| \$1: OPEN_CLS_LOCKING | Locking Clause (MS) | EX_OPEN_LOCKING NULL |
| \$2: OPEN_CLS_TYPE | Open Type | IOCX_INPUT IOCX_OUTPUT IOCX_I_O IOCX_EXTEND |
| \$3: OPEN_CLS_FILE | Open File List | EX_LIST (EX_OPEN_FILE_NAME) |

| | | |
|----------------------------|---------------------|--|
| EX_OPEN_LOCKING | Locking Clause (MS) | LOCKING IS \$1 |
| \$1: EX_OPEN_LOCKING__MODE | Lock Mode | FCX_MANUAL FCX_AUTOMATIC FCX_EXCLUSIVE |

| | | |
|---------------------|------------------|---|
| EX_OPEN_FILE_NAME | Open File Phrase | \$1 { \$2? \$2 } |
| \$1: OPEN_FILE_NAME | File Name | NAMED_OBJ_USE |
| \$2: OPEN_FILE_OPTS | Open Option | IOCX_REVERSED IOCX_WITH_LOCK IOCX_WITH_NO_REWIND NULL |

| | | |
|------------|-------|-------|
| IOCX_INPUT | Input | INPUT |
|------------|-------|-------|

| | | |
|-------------|--------|--------|
| IOCX_OUTPUT | Output | OUTPUT |
|-------------|--------|--------|

| | | |
|----------|-----|-----|
| IOCX_I_O | I-O | I-O |
|----------|-----|-----|

| | | |
|-------------|--------|--------|
| IOCX_EXTEND | Extend | EXTEND |
|-------------|--------|--------|

| | | |
|---------------|----------|----------|
| IOCX_REVERSED | Reversed | REVERSED |
|---------------|----------|----------|

10.6.41 PERFORM statement

| | | |
|---------------------|-----------------------------|--|
| STMT_PERFORM_PROC | Perform Procedure Statement | PERFORM \$1{\$2? \$2} |
| \$1: PRFP_PROC_NAME | Procedure Name Range | EX_PROC_RANGE |
| \$2: PRFP_REPEATERS | Repeater Options | EX_PERF_TIMES EX_PERF_UNTIL EX_PERF_VARYING NULL |

| | | |
|----------------------|-----------------|----------------------|
| EX_PROC_RANGE | Proc Name Range | \$1{\$2?%THRU \$2} |
| \$1: PROC_RANGE_FROM | From | NAMED_OBJ_USE |
| \$2: PROC_RANGE_TO | To | NAMED_OBJ_USE NULL |

| | | |
|-----------------------|-------------------------|--|
| STMT_PERFORM_BLOCK | Perform Block Statement | PERFORM \$1{\$2?\n\t\$2\b}-\${3?\n\$3} |
| \$1: PRFB_REPEATERS | Repeater Options | EX_PERF_TIMES EX_PERF_UNTIL EX_PERF_VARYING NULL |
| \$2: PRFB_STATEMENTS | Statements | STMT_LIST (statement) |
| \$3: PRFB_END_PERFORM | End-Perform | EX_END_PERFORM NULL |

| | | |
|--------------------------------|----------------|-------------|
| EX_PERF_TIMES | Times Repeater | \$1[TIMES] |
| \$1: EX_PERF_TIMES_HOW_MANY | How Many | ident-liter |

| | | |
|----------------------|-------------------|----------------------------------|
| EX_PERF_UNTIL | Until Repeater | { \$1? \$1 } { \$2=UNTIL \$2 } |
| \$1: PERF_UNTIL_TEST | Test Before/After | EX_PERF_WITH_TEST NULL |
| \$2: PERF_UNTIL_COND | Condition | condition |

| | | |
|----------------------------|--------------------------|-------------------------|
| EX_PERF_WITH_TEST | Test Before/After Phrase | TEST \$1 |
| \$1: EX_PERF_WITH_TEST__BA | Before/After | EX_BEFORE EX_AFTER |

| | | |
|----------------------|-------------------|--|
| EX_PERF_VARYING | Varying Repeater | { \$1? \$1 } { \$2=VARYING \$2 } |
| \$1: PERF_VARNG_TEST | Test Before/After | EX_PERF_WITH_TEST NULL |
| \$2: PERF_VARNG_VARS | Variation List | EX_PERF_VARN_LIST (EX_PERF_VARIATION) |

| | | |
|-------------------|----------------|---------|
| EX_PERF_VARN_LIST | Variation List | @&AFTER |
|-------------------|----------------|---------|

| | | |
|---------------------|-----------------|--|
| EX_PERF_VARIATION | Variation | \$1{\$2= FROM \$2}{\$3= BY \$3}{\$4= UNTIL |
| \$1: PERF_VARN_VAR | Variable | identifier |
| \$2: PERF_VARN_FROM | From | ident-liter |
| \$3: PERF_VARN_TO | To | ident-liter |
| \$4: PERF_VARN_COND | Until Condition | condition |

| | | |
|----------------|-------------|-------------|
| EX_END_PERFORM | End-Perform | END-PERFORM |
|----------------|-------------|-------------|

10.6.42 PURGE statement

| | | |
|---------------------|-----------------|---------------|
| STMT_PURGE | Purge Statement | PURGE \$1 |
| \$1: STMT_PURGE__CD | CD name | NAMED_OBJ_USE |

10.6.43 READ statement

| | | |
|----------------------------|------------------------------|--|
| STMT_READ | Read Statement | READ \$1{\$2? \$2} RECORD{\$3? \$3}{\$4? \$4} |
| \$1: READ_FILE_NAME | File Name | NAMED_OBJ_USE |
| \$2: READ_NEXT_PREV | Next/Previous | EX_READ_NEXT EX_READ_PREVIOUS NULL |
| \$3: READ_WANG_OPTN | Wang Option | EX_READ_WITH_HOLD EX_READ_MODIFIABLE EX_READ_ALTERED NULL |
| \$4: READ_INT0 | Into What | EX_READ_INT0 NULL |
| \$5: READ_LOCK_OPTN | Lock Option | EX_READ_LOCK EX_READ_KEPT_LOCK EX_READ_NO_LOCK EX_READ_IGNORE_LOCK EX_READ_WAIT NULL |
| \$6: READ_KEY | Key | EX_READ_KEY_IS NULL |
| \$7: READ_WANG_TIMEOUT | Wang Timeout | EX_WANG_TIMEOUT NULL |
| \$8: READ_ON_EXCEPTION | At End / Invalid Key | EX_ON_AT_END EX_ON_INVALID_KEY NULL |
| \$9: READ_NOT_ON_EXCEPTION | Not At End / Not Invalid Key | EX_NOT_ON_AT_END EX_NOT_ON_INVALID_KEY NULL |
| \$10: READ_END_READ | End-Read | EX_END_READ NULL |

| | | |
|--------------|------|------|
| EX_READ_NEXT | Next | NEXT |
|--------------|------|------|

| | | |
|------------------|----------|----------|
| EX_READ_PREVIOUS | Previous | PREVIOUS |
|------------------|----------|----------|

| | | |
|--------------------------|-------------|------------|
| EX_READ_INT0 | Into Clause | INTO \$1 |
| \$1: EX_READ_INT0__IDENT | Into What | identifier |

| | | |
|--------------------------|------------|-------------|
| EX_READ_KEY_IS | Key Clause | KEY IS \$1 |
| \$1: EX_READ_KEY_IS__KEY | Key | REC_KEY_USE |

| | | |
|-------------------------|----------------|--|
| REC_KEY_USE | Record Key Use | \$1 |
| \$1: REC_KEY_USE__COMPS | Key Components | FCX_REC_KEY_COMP_LIST (NAMED_OBJ_USE) |

| | | |
|--------------|-----------|-----------|
| EX_READ_LOCK | With Lock | WITH LOCK |
|--------------|-----------|-----------|

| | | |
|-------------------|----------------|----------------|
| EX_READ_KEPT_LOCK | With Kept Lock | WITH KEPT LOCK |
|-------------------|----------------|----------------|

| | | |
|-----------------|--------------|--------------|
| EX_READ_NO_LOCK | With No Lock | WITH NO LOCK |
|-----------------|--------------|--------------|

| | | |
|---------------------|------------------|------------------|
| EX_READ_IGNORE_LOCK | With Ignore Lock | WITH IGNORE LOCK |
|---------------------|------------------|------------------|

| | | |
|--------------|-----------|-----------|
| EX_READ_WAIT | With Wait | WITH WAIT |
|--------------|-----------|-----------|

| | | |
|--------------------|------------|-----------------------|
| EX_ON_AT_END | At End | AT END{\$1?\n\t\$1\b} |
| \$1: ONATEND_STMTS | Statements | STMT_LIST (statement) |

| | | |
|---------------------------------|------------|---------------------------|
| EX_NOT_ON_AT_END | Not At End | NOT AT END{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_AT_END__STMTS | Statements | STMT_LIST (statement) |

| | | |
|-------------|----------|----------|
| EX_END_READ | End-Read | END-READ |
|-------------|----------|----------|

10.6.44 READY TRACE statement

| | | |
|------------------|-----------------------|-------------|
| STMT_READY_TRACE | Ready Trace Statement | READY TRACE |
|------------------|-----------------------|-------------|

10.6.45 RECEIVE statement

| | | |
|--------------------------|-------------------|---|
| STMT_RECEIVE | Receive Statement | RECEIVE \$1{\$2= \$2}{\$3= INTO \$3}{\$4? \ |
| \$1: RECEIVE_CD | CD name | NAMED_OBJ_USE |
| \$2: RECEIVE_MSGSEG | Message/Segment | EX_RECV_MESSAGE EX_RECV_SEGMENT |
| \$3: RECEIVE_INT0 | Into What | identifier |
| \$4: RECEIVE_ONNODATA | On No Data | EX_ON_NO_DATA NULL |
| \$5: RECEIVE_NOTONNODATA | Not On No Data | EX_NOT_ON_NO_DATA NULL |
| \$6: RECEIVE_END_RECEIVE | On End-Receive | EX_END_RECEIVE NULL |

| | | |
|-----------------|---------|---------|
| EX_RECV_MESSAGE | Message | MESSAGE |
|-----------------|---------|---------|

| | | |
|-----------------|---------|---------|
| EX_RECV_SEGMENT | Segment | SEGMENT |
|-----------------|---------|---------|

| | | |
|---------------------|------------|------------------------|
| EX_ON_NO_DATA | On No Data | NO DATA{\$1?\n\t\$1\b} |
| \$1: ONNODATA_STMTS | Statements | STMT_LIST (statement) |

| | | |
|----------------------------------|----------------|--------------------------|
| EX_NOT_ON_NO_DATA | Not On No Data | WITH DATA{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_NO_DATA__STMTS | Statements | STMT_LIST (statement) |

| | | |
|----------------|-------------|-------------|
| EX_END_RECEIVE | End-Receive | END-RECEIVE |
|----------------|-------------|-------------|

10.6.46 RELEASE statement

| | | |
|---------------------|-------------------|------------------------|
| STMT_RELEASE | Release Statement | RELEASE \$1{\$2? \$2} |
| \$1: RELEASE_RECORD | Record Name | NAMED_OBJ_USE |
| \$2: RELEASE_FROM | From | EX_RELEASE_FROM NULL |

| | | |
|-----------------------------|--------------|------------|
| EX_RELEASE_FROM | Release From | FROM \$1 |
| \$1: EX_RELEASE_FROM__IDENT | From What | identifier |

10.6.47 RESET-TRACE statement

| | | |
|------------------|-----------------------|-------------|
| STMT_RESET_TRACE | Reset Trace Statement | RESET TRACE |
|------------------|-----------------------|-------------|

10.6.48 RETURN statement

| | | |
|---------------------------------|------------------|---|
| STMT_RETURN | Return Statement | RETURN \$1 RECORD{\$2? \$2}{\$3?\n\$3}{\$4? |
| \$1: RETURN_FILE | File Name | NAMED_OBJ_USE |
| \$2: RETURN_INTO | Into What | EX_READ_INTO NULL |
| \$3: RETURN_ON_EXCEPTION | At End | EX_ON_AT_END NULL |
| \$4: RETURN_NOT_ON_EXCEPTION | Not At End | EX_NOT_ON_AT_END NULL |
| \$5: RETURN_END_RETURN | End-Return | EX_END_RETURN NULL |

| | | |
|---------------|------------|------------|
| EX_END_RETURN | End-Return | END-RETURN |
|---------------|------------|------------|

10.6.49 REWRITE statement

| | | |
|----------------------------------|-------------------|---|
| STMT_REWRITE | Rewrite Statement | REWRITE \$1{\$2? \$2}{\$3? \$3}{\$4?\n\$4}{ |
| \$1: REWRITE.RECORD_NAME | Record Name | NAMED_OBJ_USE |
| \$2: REWRITE.FROM | From What | EX_WRITE_FROM NULL |
| \$3: REWRITE.WANG_AFTER | Wang After | EX_AFTER_ALARM EX_AFTER_SET_CURSOR EX_AFTER_ROLL_DOWN EX_AFTER_ROLL_UP EX_AFTER_ERASE_PROTECT EX_AFTER_ERASE_MODIFY NULL |
| \$4: REWRITE.ON_EXCEPTION | Invalid Key | EX_ON_INVALID_KEY NULL |
| \$5: REWRITE.NOT_ON_EXCEPTION | Not Invalid Key | EX_NOT_ON_INVALID_KEY NULL |
| \$6: REWRITE.END_RWRITE | End-Rewrite | EX_END_REWRITE NULL |

| | | |
|----------------|-------------|-------------|
| EX_END_REWRITE | End-Rewrite | END-REWRITE |
|----------------|-------------|-------------|

10.6.50 ROLLBACK statement

| | | |
|---------------|--------------------|----------|
| STMT_ROLLBACK | Rollback Statement | ROLLBACK |
|---------------|--------------------|----------|

10.6.51 SEARCH statement

| | | |
|------------------------|------------------|--|
| STMT_SEARCH | Search Statement | SEARCH \$1{\$2? \$2}{\$3?\n\$3}{\$4?\n\$4}{ |
| \$1: SEARCH.TABLE | Table | NAMED_OBJ_USE |
| \$2: SEARCH.VARYING | Varying Phrases | EX_SEARCH_VARYING NULL |
| \$3: SEARCH.AT_END | At End | EX_SEARCH_AT_END NULL |
| \$4: SEARCH.WHEN | When Phrases | EX_SEARCH_WHEN_PHRASE_LIST (EX_SEARCH_WHEN_PHRASE) NULL |
| \$5: SEARCH.END_SEARCH | End-Search | EX_END_SEARCH NULL |

| | | |
|----------------------------|------------------|--|
| STMT_SEARCH_ALL | Search Statement | SEARCH ALL \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{ |
| \$1: SEARCH.ALL.TABLE | Table | NAMED_OBJ_USE |
| \$2: SEARCH.ALL.AT_END | At End | EX_SEARCH_AT_END NULL |
| \$3: SEARCH.ALL.WHEN | When Phrase | EX_SEARCH_WHEN_PHRASE |
| \$4: SEARCH.ALL.END_SEARCH | End-Search | EX_END_SEARCH NULL |

| | | |
|----------------------------------|----------------|-------------|
| EX_SEARCH_VARYING | Varying Clause | VARYING \$1 |
| \$1: EX_SEARCH_VARYING__IDENT | Identifier | identifier |

| | | |
|-------------------------|------------|-----------------------|
| EX_SEARCH_AT_END | At End | AT END{\$1?\n\t\$1\b} |
| \$1: SEARCH.ATEND_STMTS | Statements | STMT_LIST (statement) |

| | | |
|----------------------------|-------------------------|--------|
| EX_SEARCH_WHEN_PHRASE_LIST | Search When Phrase List | \r@\\n |
|----------------------------|-------------------------|--------|

| | | |
|------------------------|--------------------|------------------------------|
| EX_SEARCH_WHEN_PHRASE | Search When Phrase | WHEN \$1{\$2?\n\t\$2\b} |
| \$1: SEARCH_WHEN_COND | Condition | condition |
| \$2: SEARCH_WHEN_STMTS | Statements | STMT_LIST (statement) NULL |

| | | |
|---------------|------------|------------|
| EX_END_SEARCH | End-Search | END-SEARCH |
|---------------|------------|------------|

10.6.52 SEND statement

| | | |
|---------------------|----------------|--|
| STMT_SEND | Send Statement | SEND \$1{\$2? \$2}{\$3? WITH \$3}{\$4? \$4} |
| \$1: SEND_CD | CD name | NAMED_OBJ_USE |
| \$2: SEND_FROM | From | EX_WRITE_FROM NULL |
| \$3: SEND_WITH | With | EX_SEND_ESI EX_SEND_EMI EX_SEND_EGI NULL |
| \$4: SEND_ADVANCING | With | IOCX_ADVANCING NULL |
| \$5: SEND_REPLACING | With | EX_SEND_REPLACING_LINE NULL |

| | | |
|-------------|-----|-----|
| EX_SEND_ESI | Esi | ESI |
|-------------|-----|-----|

| | | |
|-------------|-----|-----|
| EX_SEND_EMI | Emi | EMI |
|-------------|-----|-----|

| | | |
|-------------|-----|-----|
| EX_SEND_EGI | Egi | EGI |
|-------------|-----|-----|

| | | |
|------------------------|----------------|----------------|
| EX_SEND_REPLACING_LINE | Replacing Line | REPLACING LINE |
|------------------------|----------------|----------------|

10.6.53 SERVICE RELOAD statement

| | | |
|---------------------------------|-------------------|--------------------|
| STMT_SERVICE_RELOAD | Service Statement | SERVICE RELOAD \$1 |
| \$1: STMT_SERVICE_RELOAD__IDENT | Identifier | identifier |

10.6.54 SET statement

| | | |
|---------------------------|-----------------|----------------------------|
| STMT_SET_TO | Set Statement | SET \$1 |
| \$1: STMT_SET_TO__CLAUSES | Set Clause List | EX_LIST (EX_SET_TO_CLAUSE) |

| | | |
|--------------------|--------------|---|
| EX_SET_TO_CLAUSE | Set Clause | \$1{\$2= TO \$2} |
| \$1: SETTOCL_DESTS | Destinations | IDENT_LIST (identifier) |
| \$2: SETTOCL_SRC | Source | ident-liter EX_ON EX_OFF EX_TRUE EX_FALSE EX_PTR_ENTRY |

| | | |
|---------------------|---------------|----------------------------------|
| STMT_SET_UP_DOWN_BY | Set Statement | SET \$1{\$2= \$2}{\$3= BY \$3} |
| \$1: SETUD_DESTS | Destinations | IDENT_LIST (identifier) |
| \$2: SETUD_UPDOWN | Up/Down | EX_SET_UP_BY EX_SET_DOWN_BY |
| \$3: SETUD_BYWHAT | By What | ident-liter |

| | | |
|--------------|----|----|
| EX_SET_UP_BY | Up | UP |
|--------------|----|----|

| | | |
|----------------|------|------|
| EX_SET_DOWN_BY | Down | DOWN |
|----------------|------|------|

| | | |
|-------------------|---------------|---------------------------------|
| STMT_SET_ADDRESS | Set Statement | SET ADDRESS OF \$1{\$2= TO \$2} |
| \$1: SETADDR_DEST | Address Of | identifier |
| \$2: SETADDR_SRC | Address Of | ident-liter |

| | | |
|-------|----|----|
| EX_ON | On | ON |
|-------|----|----|

| | | |
|--------|-----|-----|
| EX_OFF | Off | OFF |
|--------|-----|-----|

| | | |
|---------|------|------|
| EX_TRUE | True | TRUE |
|---------|------|------|

| | | |
|----------|-------|-------|
| EX_FALSE | False | FALSE |
|----------|-------|-------|

| | | |
|--------|-----|-----|
| EX_ANY | Any | ANY |
|--------|-----|-----|

| | | |
|--------------------------|------------------|-------------|
| EX_PTR_ENTRY | Address Of Entry | ENTRY \$1 |
| \$1: EX_PTR_ENTRY__IDENT | Entry Name | ident-liter |

10.6.55 SORT statement

| | | |
|-----------------------|---------------------------|--|
| STMT_SORT | Sort Statement | SORT \$1{\$2? \$2}{\$3? \$3}{\$4? \$4}{\$5? \$5} |
| \$1: SORT_NAME | File/Table Name | NAMED_OBJ_USE |
| \$2: SORT_KEYS | Keys | EX_LIST (EX_SORT_KEY) NULL |
| \$3: SORT_DUPS | Duplicates | EX_WITH_DUPS_IN_ORDER NULL |
| \$4: SORT_COLLSEQ | Collating Sequence | EX_SORT_COLL_SEQCE NULL |
| \$5: SORT_INPUT_PROC | Input Procedure / Using | EX_INPUT_PROC_IS EX_SORT_USING NULL |
| \$6: SORT_OUTPUT_PROC | Output Procedure / Giving | EX_OUTPUT_PROC_IS EX_GIVING NULL |

| | | |
|----------------------|----------------------|---------------------------------|
| EX_SORT_KEY | Sort Key | ON \$1 KEY{\$2? \$2} |
| \$1: SORTKEY_ASCDESC | Ascending/Descending | EX_ASCENDING EX_DESCENDING |
| \$2: SORTKEY_COMPS | Key Components | EX_LIST (NAMED_OBJ_USE) |

| | | |
|--------------------------------------|--------------------|---------------------------|
| EX_SORT_COLL_SEQCE | Collating Sequence | COLLATING SEQUENCE IS \$1 |
| \$1: EX_SORT_COLL_SEQCE__ALPHABET | Alphabet | NAMED_OBJ_USE |

| | | |
|-----------------------------|-----------------|------------------------|
| EX_INPUT_PROC_IS | Input Procedure | INPUT PROCEDURE IS \$1 |
| \$1: EX_INPUT_PROC_IS__PROC | Procedure | NAMED_OBJ_USE |

| | | |
|----------------------------|-------------------------|-------------------------|
| EX_SORT_USING | Using File-Names Clause | USING \$1 |
| \$1: EX_SORT_USING__PARAMS | File-Name List | EX_LIST (NAMED_OBJ_USE) |

| | | |
|---------------------------------|------------------|-------------------------|
| EX_OUTPUT_PROC_IS | Output Procedure | OUTPUT PROCEDURE IS \$1 |
| \$1: EX_OUTPUT_PROC_IS__PROC | Procedure | NAMED_OBJ_USE |

| | | |
|-----------------------|-----------|-------------------------|
| EX_GIVING | Giving | GIVING \$1 |
| \$1: EX_GIVING__NAMES | Procedure | EX_LIST (NAMED_OBJ_USE) |

| | | |
|-----------------------|--------------------------|--------------------------|
| EX_WITH_DUPS_IN_ORDER | With Duplicates In Order | WITH DUPLICATES IN ORDER |
|-----------------------|--------------------------|--------------------------|

10.6.56 START statement

| | | |
|---------------------------------|-----------------|--|
| STMT_START | Start Statement | START{\$1? \$1}{\$2? 2}{\$3? 3}{\$4? 4}\n\$ |
| \$1: STMT_START__FILE | File Name | NAMED_OBJ_USE |
| \$2: STMT_START__KEY | Key | relation |
| \$3: STMT_START__WITH | With Phrase | EX_START_WITH_SIZE EX_START_WITH_REV_ORDER NULL |
| \$4: STMT_START__INV_KEY | Invalid Key | EX_ON_INVALID_KEY NULL |
| \$5: STMT_START__NOT_INV_KEY | Not Invalid Key | EX_NOT_ON_INVALID_KEY NULL |
| \$6: STMT_START__END_START | End-Start | EX_END_START NULL |

| | | |
|-------------------------------------|----------------------|-------------------------------------|
| START_KEY_FIRST_LAST | First/Last Start Key | \$1 KEY{\$2= IS \$2} |
| \$1: START_FIRST_LAST_FIRST_LAST | First or Last | START_KEY_FIRST START_KEY_LAST |
| \$2: START_FIRST_LAST_KEY | Key | REC_KEY_USE |

| | | |
|-----------|-----|-----|
| START_KEY | Key | KEY |
|-----------|-----|-----|

| | | |
|-----------------|-----------|-------|
| START_KEY_FIRST | First Key | FIRST |
|-----------------|-----------|-------|

| | | |
|----------------|----------|------|
| START_KEY_LAST | Last Key | LAST |
|----------------|----------|------|

| | | |
|-----------------------------------|-------------|---------------|
| EX_START_WITH_SIZE | Size Phrase | WITH SIZE \$1 |
| \$1: EX_START_WITH_SIZE__VALUE | Value | ident-liter |

| | | |
|-------------------------|---------------------|---------------------|
| EX_START_WITH_REV_ORDER | With Reversed Order | WITH REVERSED ORDER |
|-------------------------|---------------------|---------------------|

| | | |
|--------------|-----------|-----------|
| EX_END_START | End-Start | END-START |
|--------------|-----------|-----------|

10.6.57 STOP statement

| | | |
|-------------------|----------------|------------------------------|
| STMT_STOP | Stop Statement | STOP{\$1? 1}{\$2? 2}{\$3? 3} |
| \$1: STOP_RUN | Run | EX_RUN NULL |
| \$2: STOP_LITERAL | Literal | literal NULL |
| \$3: STOP_GIVING | Giving | EX_EXIT_GIVING NULL |

| | | |
|--------|-----|-----|
| EX_RUN | Run | RUN |
|--------|-----|-----|

10.6.58 STRING statement

| | | |
|------------------------|--------------------|---|
| STMT_STRING | String Statement | STRING \$1{\$2= INTO \$2}{\$3? \$3}{\$4? \n |
| \$1: STRING_PHRASES | String Phrase List | EX_LIST (EX_STR_DELIMITED_BY) |
| \$2: STRING_INT0 | Into What | identifier |
| \$3: STRING_POINTER | With Pointer | EX_STR_WITH_POINTER NULL |
| \$4: STRING_ONOVER | On Overflow | EX_ON_STRING_OVERFLOW NULL |
| \$5: STRING_NOTONOVER | Not On Overflow | EX_NOT_ON_STRING_OVERFLOW NULL |
| \$6: STRING_END_STRING | End-String | EX_END_STRING NULL |

| | | |
|------------------------------------|--------------|-------------|
| EX_STR_WITH_POINTER | With Pointer | POINTER \$1 |
| \$1: EX_STR_WITH_POINTER__IDENT | Identifier | identifier |

| | | |
|---------------------|---------------------|------------------------------|
| EX_STR_DELIMITED_BY | Delimited By Phrase | \$1{\$2= DELIMITED BY \$2} |
| \$1: STRDELBY_SENDS | Sending Items | EX_LIST (ident-liter) |
| \$2: STRDELBY_DELTR | Delimiter | EX_STR_SIZE ident-liter |

| | | |
|-------------|------|------|
| EX_STR_SIZE | Size | SIZE |
|-------------|------|------|

| | | |
|-----------------------|--------------------|----------------------------|
| EX_ON_STRING_OVERFLOW | On String Overflow | ON OVERFLOW{\$1?\n\t\$1\b} |
| \$1: ONSTROVFL_STMTS | Statements | STMT_LIST (statement) |

| | | |
|--|------------------------|--------------------------------|
| EX_NOT_ON_STRING_OVERFLOW | Not On String Overflow | NOT ON OVERFLOW{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_STRING_OVERFLOW__STMTS | Statements | STMT_LIST (statement) |

| | | |
|---------------|------------|------------|
| EX_END_STRING | End-String | END-STRING |
|---------------|------------|------------|

10.6.59 SUBTRACT statement

| | | |
|----------------------------|--------------------|---|
| STMT_SUBTRACT | Subtract Statement | SUBTRACT \$1{\$2?\n\$2}{\$3?\n\$3}{\$4? \n\$ |
| \$1: SUBTRACT_FORMAT | Subtract Format | EX_ARITH_SUB_FROM EX_ARITH_SUB_FROM_GIVING EX_ARITH_SUB_CORR_FROM |
| \$2: SUBTRACT_ONSIZE | On Size Error | EX_ON_SIZE_ERROR NULL |
| \$3: SUBTRACT_NOTONSIZE | Not On Size Error | EX_NOT_ON_SIZE_ERROR NULL |
| \$4: SUBTRACT_END_SUBTRACT | End-Subtract | EX_END_SUBTRACT NULL |

| | | |
|------------------------|----------------------|------------------------------|
| EX_ARITH_SUB_FROM | Subtract From Format | \$1{\$2= FROM \$2} |
| \$1: SUBFROM_WHAT | Subtract What | EX_LIST (ident-liter) |
| \$2: SUBFROM_FROM_WHAT | Subtract From What | EX_LIST (identifier-rounded) |

| | | |
|--------------------------|----------------------|-------------------------------------|
| EX_ARITH_SUB_FROM_GIVING | Subtract From Giving | \$1{\$2= FROM \$2}{\$3= GIVING \$3} |
| \$1: SUBFROMGIVE_ARG1 | Args1 | EX_LIST (ident-liter) |
| \$2: SUBFROMGIVE_ARG2 | Arg2 | ident-liter |
| \$3: SUBFROMGIVE_DESTS | Destinations | EX_LIST (identifier-rounded) |

| | | |
|------------------------|------------------------|----------------------------------|
| EX_ARITH_SUB_CORR_FROM | Subtract Corresponding | CORRESPONDING \$1{\$2= FROM \$2} |
| \$1: SUBCORR_WHAT | Subtract What | identifier |
| \$2: SUBCORR_FROMWHAT | From What | identifier-rounded |

| | | |
|-----------------|--------------|--------------|
| EX_END_SUBTRACT | End-Subtract | END-SUBTRACT |
|-----------------|--------------|--------------|

10.6.60 SUPPRESS PRINTING statement

| | | |
|------------------------|--------------------|-------------------|
| STMT_SUPPRESS_PRINTING | Suppress Statement | SUPPRESS PRINTING |
|------------------------|--------------------|-------------------|

10.6.61 TERMINATE statement

| | | |
|--------------------------------|---------------------|-------------------------|
| STMT_TERMINATE | Terminate Statement | TERMINATE \$1 |
| \$1: STMT_TERMINATE_REPORTS | Reports | EX_LIST (NAMED_OBJ_USE) |

10.6.62 TRANSFORM statement

| | | |
|---------------------|---------------------|---|
| STMT_TRANSFORM | Transform Statement | TRANSFORM \$1{\$2= FROM \$2}{\$3= TO \$3} |
| \$1: TRANSFORM_WHAT | What | identifier |
| \$2: TRANSFORM_FROM | From | ident-liter |
| \$3: TRANSFORM_TO | To | ident-liter |

10.6.63 UNLOCK statement

| | | |
|------------------------|------------------|-------------------------|
| STMT_UNLOCK | Unlock Statement | UNLOCK \$1[RECORDS] |
| \$1: STMT_UNLOCK_FILES | Files | EX_LIST (NAMED_OBJ_USE) |

10.6.64 UNDELETE statement

| | | |
|-------------------------|--------------------|------------------------|
| STMT_UNDELETE | Undelete Statement | UNDELETE \$1[RECORDS] |
| \$1: STMT_UNDELETE_FILE | File | NAMED_OBJ_USE |

10.6.65 UNSTRING statement

| | | |
|----------------------------|--------------------|--|
| STMT_UNSTRING | Unstring Statement | UNSTRING \$1{\$2? \$2}{\$3= INTO \$3}{\$4? |
| \$1: UNSTRING_WHAT | What | identifier |
| \$2: UNSTRING_DELIMITED | Delimited By | EX_UNSTR_DELIMITED_BY NULL |
| \$3: UNSTRING_INT0 | Into | EX_LIST (EX_UNSTR_INT0_PHRASE) |
| \$4: UNSTRING_POINTER | With Pointer | EX_STR_WITH_POINTER NULL |
| \$5: UNSTRING_TALLYING | Tallying | EX_UNSTR_TALLYING_IN NULL |
| \$6: UNSTRING_ONOVER | On Overflow | EX_ON_STRING_OVERFLOW NULL |
| \$7: UNSTRING_NOTONOVER | Not On Overflow | EX_NOT_ON_STRING_OVERFLOW NULL |
| \$8: UNSTRING_END_UNSTRING | End-Unstring | EX_END_UNSTRING NULL |

| | | |
|--|--------------------------|---|
| EX_UNSTR_DELIMITED_BY | Delimited By Clause | DELIMITED BY \$1 |
| \$1: EX_UNSTR_DELIMITED_BY__PHRASES | Delimited By Phrase List | EX_UNSTR_OR_LIST (ident-liter EX_UNSTR_ALL) |

| | | |
|------------------|---------------------|-------|
| EX_UNSTR_OR_LIST | Delimited By Phrase | @ OR& |
|------------------|---------------------|-------|

| | | |
|--------------------------|------------------------|-------------|
| EX_UNSTR_ALL | Unstring All Qualifier | ALL \$1 |
| \$1: EX_UNSTR_ALL__VALUE | Value | ident-liter |

| | | |
|---------------------------|----------------------|------------------------------|
| EX_UNSTR_INT0_PHRASE | Unstring Into Phrase | \$1{\$2? \$2}{\$3? \$3} |
| \$1: UNSTRING_IP_INTOWHAT | Into What | identifier |
| \$2: UNSTRING_IP_DELIM | Delimiter | EX_UNSTR_DELIMITER_IN NULL |
| \$3: UNSTRING_IP_COUNT | Count | EX_UNSTR_COUNT_IN NULL |

| | | |
|--------------------------------------|--------------|------------------|
| EX_UNSTR_DELIMITER_IN | Delimiter In | DELIMITER IN \$1 |
| \$1: EX_UNSTR_DELIMITER_IN__IDENT | Identifier | identifier |

| | | |
|----------------------------------|------------|--------------|
| EX_UNSTR_COUNT_IN | Count In | COUNT IN \$1 |
| \$1: EX_UNSTR_COUNT_IN__IDENT | Identifier | identifier |

| | | |
|-------------------------------------|--------------------|-----------------|
| EX_UNSTR_TALLYING_IN | Tallying In Clause | TALLYING IN \$1 |
| \$1: EX_UNSTR_TALLYING_IN__IDENT | Identifier | identifier |

| | | |
|-----------------|--------------|--------------|
| EX_END_UNSTRING | End-Unstring | END-UNSTRING |
|-----------------|--------------|--------------|

10.6.66 USE statement

| | | |
|----------------------|---------------------|--|
| STMT_USE_AFTER | Use After Statement | USE{\$1? \$1}{\$2= AFTER STANDARD \$2}{\$3= |
| \$1: USEAFTER_GLOBAL | Global | EX_USE_GLOBAL NULL |
| \$2: USEAFTER_EVENT | After What Event | IOCX_ERROR IOCX_BEGINNING IOCX_ENDING |
| \$3: USEAFTER_ONWHAT | On What | EX_LIST (NAMED_OBJ_USE) IOCX_INPUT IOCX_OUTPUT IOCX_I_O IOCX_EXTEND IOCX_SHARED IOCX_SPECIAL_INPUT |
| \$4: USEAFTER_GIVING | Giving Clause | EX_USE_GIVING NULL |

| | | |
|---------------|--------|--------|
| EX_USE_GLOBAL | Global | GLOBAL |
|---------------|--------|--------|

| | | |
|------------|-------|-------|
| IOCX_ERROR | Error | ERROR |
|------------|-------|-------|

| | | |
|--------------------------|------------------|---|
| IOCX_BEGINNING | Beginning Clause | BEGINNING \$1 |
| \$1: IOCX_BEGINNING__FRU | File/Reel/Unit | IOCX_FILE IOCX_REEL IOCX_UNIT |

| | | |
|-----------------------|----------------|---|
| IOCX_ENDING | Ending Clause | ENDING \$1 |
| \$1: IOCX_ENDING__FRU | File/Reel/Unit | IOCX_FILE IOCX_REEL IOCX_UNIT |

| | | |
|-----------|------|------|
| IOCX_FILE | File | FILE |
|-----------|------|------|

| | | |
|-----------|------|------|
| IOCX_REEL | Reel | REEL |
|-----------|------|------|

| | | |
|-----------|------|------|
| IOCX_UNIT | Unit | UNIT |
|-----------|------|------|

| | | |
|--------------------------|-------------------|------------|
| EX_USE_GIVING | Use Giving Clause | GIVING \$1 |
| \$1: EX_USE_GIVING__WHAT | What | EX_USE_ARG |

| | | |
|------------------|---------------------|-------------------|
| EX_USE_ARG | Use Giving Argument | \$1{\$2? \$2} |
| \$1: USEARG_ARG1 | First Data Name | identifier |
| \$2: USEARG_ARG2 | Second Data Name | identifier NULL |

| | | |
|---------------------------|--------------------------------|--|
| STMT_USE_BEFORE_REPORTING | Use Before Reporting Statement | USE{\$1? GLOBAL}{\$2= BEFORE REPORTING |
| \$1: USEBEFREP_GLOBAL | Identifier | EX_USE_GLOBAL NULL |
| \$2: USEBEFREP_IDENT | Identifier | identifier |

| | | |
|--------------------------------------|---------------|---|
| STMT_USE_FOR_DEBUGGING | Use Statement | USE FOR DEBUGGING ON \$1 |
| \$1: STMT_USE_FOR_DEBUGGING_ITEMS | Item List | EX_LIST (identifier EX_DEBUG_ALL_REFERENCES_OF EX_DEBUG_ALL_PROCEDURES) |

| | | |
|--|-------------------------|-----------------------|
| EX_DEBUG_ALL_REFERENCES_OF | Debug All References Of | ALL REFERENCES OF \$1 |
| \$1: EX_DEBUG_ALL_REFERENCES_OF_IDENT | Identifier | identifier |

| | | |
|-------------------------|----------------------|----------------|
| EX_DEBUG_ALL_PROCEDURES | Debug All Procedures | ALL PROCEDURES |
|-------------------------|----------------------|----------------|

10.6.67 WRITE statement

| | | |
|--------------------------------|------------------------------|--|
| STMT_WRITE | Write Statement | WRITE \$1{\$2? \$2}{\$3? \$3}{\$4? \$4}{\$5? |
| \$1: WRITE_RECORD_NAME | Record Name | NAMED_OBJ_USE |
| \$2: WRITE_FROM | From What | EX_WRITE_FROM NULL |
| \$3: WRITE_ADV_POS | Before/After Advng/Posng | IOCX_ADVANCING IOCX_POSITIONING NULL |
| \$4: WRITE_WANG_TIMEOUT | Wang Timeout | EX_WANG_TIMEOUT NULL |
| \$5: WRITE_ON_EXCEPTION | At End / Invalid Key | EX_ON_AT_EOP EX_ON_INVALID_KEY NULL |
| \$6: WRITE_NOT_ON_EXCEPTION | Not At End / Not Invalid Key | EX_NOT_ON_AT_EOP EX_NOT_ON_INVALID_KEY NULL |
| \$7: WRITE_END_WRITE | End-Write | EX_END_WRITE NULL |

| | | |
|---------------------------------------|-------------|-------------|
| EX_WRITE_FROM | From Clause | FROM \$1 |
| \$1: EX_WRITE_FROM_WRITE_FROM_WHAT | From What | ident-liter |

| | | |
|-----------------|------------------|-------------------------|
| IOCX_ADVANCING | Advancing Clause | \$1{\$2= ADVANCING \$2} |
| \$1: ADVNG_BA | Before/After | EX_BEFORE EX_AFTER |
| \$2: ADVNG_WHAT | Lines/Page | EX_LINES EX_PAGE |

| | | |
|------------------|--------------------|---------------------------|
| IOCX_POSITIONING | Positioning Clause | \$1{\$2= POSITIONING \$2} |
| \$1: POSNG_BA | Before/After | EX_BEFORE EX_AFTER |
| \$2: POSNG_WHAT | Lines | EX_LINES |

| | | |
|----------------------|--------------|-------------|
| EX_LINES | Lines Phrase | \$1[LINES] |
| \$1: EX_LINES__VALUE | Value | ident-liter |

| | | |
|---------|-------------|------|
| EX_PAGE | Page Phrase | PAGE |
|---------|-------------|------|

| | | |
|--------------------|-----------------|-------------------------------|
| EX_ON_AT_EOP | On At EndOfPage | AT END-OF-PAGE{\$1?\n\t\$1\b} |
| \$1: ONATEOP_STMTS | Statements | STMT_LIST (statement) |

| | | |
|---------------------------------|---------------------|-----------------------------------|
| EX_NOT_ON_AT_EOP | Not On At EndOfPage | NOT AT END-OF-PAGE{\$1?\n\t\$1\b} |
| \$1: EX_NOT_ON_AT_EOP__STMTS | Statements | STMT_LIST (statement) |

| | | |
|--------------|-----------|-----------|
| EX_END_WRITE | End-Write | END-WRITE |
|--------------|-----------|-----------|

10.7 Common Expression Subtrees

| | | |
|----------------------|-----------------------------|--|
| • identifier-rounded | Identifier Possibly Rounded | identifier IDENT_ROUNDED (identifier) |
|----------------------|-----------------------------|--|

| | | |
|---------------------------|--------------------|---------------|
| IDENT_ROUNDED | Identifier Rounded | \$1[ROUNDED] |
| \$1: IDENT_ROUNDED__IDENT | Identifier | identifier |

| | | |
|------------|-----------------|-----|
| IDENT_LIST | Identifier List | @\n |
|------------|-----------------|-----|

| | | |
|---------------|-----------------------|---|
| • ident-liter | Identifier Or Literal | identifier literal INTR_FUNCTION SR_ADDRESS_OF SR_LENGTH_OF |
|---------------|-----------------------|---|

| | | |
|--------------|------------|---|
| • identifier | Identifier | REFERENCE_MOD TABLE_ELT NAMED_OBJ_USE |
|--------------|------------|---|

10.7.1 Leaf Operations

| | | |
|----------------------|----------------------|--|
| • literal | Literal | literal-1 FIG_ALL |
| FIG_ALL | All Qualifier | ALL \$1 |
| \$1: FIG_ALL_LITERAL | Literal | literal-1 |
| • literal-1 | Literal Without All | INT_NUM_CONST REAL_NUM_CONST FIG_ZERO nonnum-literal |
| • nonnum-literal | NonNumeric Literal | STR_CONST FIG_SPACE FIG_HIGH_VALUE FIG_LOW_VALUE FIG_QUOTE FIG_NULL |
| FIG_ZERO | Zero Constant | ZERO |
| FIG_SPACE | Spaces Constant | SPACES |
| FIG_HIGH_VALUE | High-Values Constant | HIGH-VALUES |
| FIG_LOW_VALUE | Low-Values Constant | LOW-VALUES |
| FIG_QUOTE | Quote Constant | QUOTE |
| FIG_NULL | Null Constant | NULL |
| STR_CONST | String Constant | |
| CHAR_STRING | Char String | |
| INT_NUM_CONST | Integer Constant | |

| | | |
|----------------|-----------------------|--|
| REAL_NUM_CONST | Real Numeric Constant | |
|----------------|-----------------------|--|

| | | |
|---------------|-------------------------|--|
| NAMED_OBJ_DEF | Named Object Definition | |
|---------------|-------------------------|--|

| | | |
|---------------------------|------------------|-----------------------|
| NAMED_OBJ_USE | Named Object Use | \$1 |
| \$1: NAMED_OBJ_USE__NAMES | Names | OF_IN_LIST (NAME_USE) |

| | | |
|----------|------------------|--|
| NAME_USE | Named Object Use | |
|----------|------------------|--|

| | | |
|------------|------------|------|
| OF_IN_LIST | Of/In List | @%OF |
|------------|------------|------|

| | | |
|---------------------------|------------------------|----------------|
| SR_ADDRESS_OF | Address Of Special Reg | ADDRESS OF \$1 |
| \$1: SR_ADDRESS_OF__IDENT | Identifier | identifier |

| | | |
|--------------------------|-----------------------|---------------|
| SR_LENGTH_OF | Length Of Special Reg | LENGTH OF \$1 |
| \$1: SR_LENGTH_OF__IDENT | Identifier | identifier |

10.7.2 Intrinsic Function

| | | |
|----------------|--------------------|---|
| INTR_FUNCTION | Intrinsic Function | FUNCTION \$1{\$2? \$2} |
| \$1: FUNC_NAME | Name | CHAR_STRING |
| \$2: FUNC_ARGS | Arguments | OP_PARENS (FUNC_ARG_LIST (arith-expr)) |

| | | |
|---------------|------------------------|-----|
| FUNC_ARG_LIST | Function Argument List | @,% |
|---------------|------------------------|-----|

10.7.3 Reference Modification

| | | |
|--------------------|------------------------|-----------------------------------|
| REFERENCE_MOD | Reference Modification | \$1{\$2= (\$2 \\\: }-{\$3?\$3}[]] |
| \$1: REFMOD_WHAT | What | TABLE_ELT NAMED_OBJ_USE |
| \$2: REFMOD_FROM | From Pos | arith-expr |
| \$3: REFMOD_LENGTH | Length | arith-expr |

10.7.4 Table Element

| | | |
|-------------------|---------------|---|
| TABLE_ELT | Table Element | \$1{\$2= (\$2)} |
| \$1: TABELT_TABLE | Table | NAMED_OBJ_USE |
| \$2: TABELT_SUBS | Subscripts | SUBSCRIPT_LIST (INT_NUM_CONST NAMED_OBJ_USE OP_ADD_IX OP_SUB_IX EX_INDEX_ALL) |

| | | |
|----------------|----------------|----|
| SUBSCRIPT_LIST | Subscript List | @& |
|----------------|----------------|----|

| | | |
|-------------------|-------------|-----------------|
| OP_ADD_IX | Add Indices | \$1 +{\$2=&\$2} |
| \$1: OPADDIX_ARG1 | Arg 1 | ident-liter |
| \$2: OPADDIX_ARG2 | Arg 2 | INT_NUM_CONST |

| | | |
|-------------------|------------------|-----------------|
| OP_SUB_IX | Subtract Indices | \$1 -{\$2=&\$2} |
| \$1: OPSUBIX_ARG1 | Arg 1 | ident-liter |
| \$2: OPSUBIX_ARG2 | Arg 2 | INT_NUM_CONST |

| | | |
|--------------|-----------|-----|
| EX_INDEX_ALL | Index All | ALL |
|--------------|-----------|-----|

10.7.5 Arithmetic Expression

| | | |
|------------------|---------------------------|--|
| • rel-arith-expr | Rel Arithmetic Expression | arith-expr START_KEY START_KEY_WANG REC_KEY_USE |
|------------------|---------------------------|--|

| | | |
|--------------|-----------------------|--|
| • arith-expr | Arithmetic Expression | ident-liter AR_ADD (arith-expr) AR_SUB (arith-expr) AR_MUL (arith-expr) AR_DIV (arith-expr) AR_EXP (arith-expr) AR_UN_PLUS AR_UN_MINUS OP_PARENS |
|--------------|-----------------------|--|

| | | |
|--------|-----|------|
| AR_ADD | Add | @ +& |
|--------|-----|------|

| | | |
|--------|----------|------|
| AR_SUB | Subtract | @ -& |
|--------|----------|------|

| | | |
|--------|----------|------|
| AR_MUL | Multiply | @ *& |
|--------|----------|------|

| | | |
|-----------------------|---------------|------------|
| AR_DIV | Divide | @ /& |
| AR_EXP | Exponent | @ **& |
| BIT_AND | Bit And | @ AND& |
| BIT_OR | Bit Or | @ OR& |
| AR_UN_MINUS | Unary Minus | - \$1 |
| \$1: AR_UN_MINUS__ARG | Arg | arith-expr |
| AR_UN_PLUS | Unary Plus | + \$1 |
| \$1: AR_UN_PLUS__ARG | Arg | arith-expr |
| STR_CONCATENATE | Concatenation | @%\&& |
| OP_PARENS | Parentheses | (\$1) |
| \$1: OP_PARENS__ARG | Arg | arith-expr |

10.7.6 Condition

| | | |
|-------------|-----------|---|
| • condition | Condition | relation CMP_NUM CMP_ALPH CMP_ALPH_LOWER CMP_ALPH_UPPER CMP_POS CMP_NEG CMP_ZER CMP_IS_CLASS CMP_NOT_NUM CMP_NOT_ALPH CMP_NOT_ALPH_LOWER CMP_NOT_ALPH_UPPER CMP_NOT_POS CMP_NOT_NEG CMP_NOT_ZER CMP_IS_NOT_CLASS CMP_ALTERED LOG_AND (condition) LOG_OR (condition) LOG_NOT |
|-------------|-----------|---|

| | | |
|------------|----------|--|
| • relation | Relation | CMP_EQ CMP_GT CMP_GE CMP_LT CMP_LE CMP_NOT_EQ CMP_NOT_GT CMP_NOT_GE CMP_NOT_LT CMP_NOT_LE |
|------------|----------|--|

| | | |
|-------------------|-------|----------------|
| CMP_EQ | Equal | \$1{\$2= =\$2} |
| \$1: CMP_EQ__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_EQ__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-------------------|--------------|-----------------|
| CMP_GT | Greater Than | \$1{\$2= > \$2} |
| \$1: CMP_GT__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_GT__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-------------------|-----------------------|------------------|
| CMP_GE | Greater Than Or Equal | \$1{\$2= >= \$2} |
| \$1: CMP_GE__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_GE__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-------------------|-----------|-----------------|
| CMP_LT | Less Than | \$1{\$2= < \$2} |
| \$1: CMP_LT__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_LT__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-------------------|--------------------|------------------|
| CMP_LE | Less Than Or Equal | \$1{\$2= <= \$2} |
| \$1: CMP_LE__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_LE__ARG2 | Arg2 | rel-arith-expr |

| | | |
|---------------------|------------|---------------|
| CMP_NUM | Is Numeric | \$1 [NUMERIC] |
| \$1: CMP_NUM__IDENT | Identifier | identifier |

| | | |
|----------------------|---------------|------------------|
| CMP_ALPH | Is Alphabetic | \$1 [ALPHABETIC] |
| \$1: CMP_ALPH__IDENT | Identifier | identifier |

| | | |
|----------------------------|---------------------|------------------------|
| CMP_ALPH_LOWER | Is Alphabetic-Lower | \$1 [ALPHABETIC-LOWER] |
| \$1: CMP_ALPH_LOWER__IDENT | Identifier | identifier |

| | | |
|----------------------------|---------------------|------------------------|
| CMP_ALPH_UPPER | Is Alphabetic-Upper | \$1 [ALPHABETIC-UPPER] |
| \$1: CMP_ALPH_UPPER__IDENT | Identifier | identifier |

| | | |
|---------------------|-------------|----------------|
| CMP_POS | Is Positive | \$1 [POSITIVE] |
| \$1: CMP_POS__IDENT | Identifier | identifier |

| | | |
|---------------------|-------------|-----------------|
| CMP_NEG | Is Negative | \$1 [NEGATIVE] |
| \$1: CMP_NEG__IDENT | Identifier | identifier |

| | | |
|---------------------|------------|-------------|
| CMP_ZER | Is Zero | \$1 [ZERO] |
| \$1: CMP_ZER__IDENT | Identifier | identifier |

| | | |
|--------------------------|-------------|------------------|
| CMP_IS_CLASS | Is Of Class | \$1{\$2= IS \$2} |
| \$1: CMP_IS_CLASS__IDENT | Identifier | identifier |
| \$2: CMP_IS_CLASS__CLASS | Identifier | NAMED_OBJ_USE |

| | | |
|-----------------------|-----------|---------------------|
| CMP_NOT_EQ | Not Equal | \$1{\$2=%NOT = \$2} |
| \$1: CMP_NOT_EQ__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_NOT_EQ__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-----------------------|------------------|---------------------|
| CMP_NOT_GT | Not Greater Than | \$1{\$2=%NOT > \$2} |
| \$1: CMP_NOT_GT__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_NOT_GT__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-----------------------|---------------------------|----------------------|
| CMP_NOT_GE | Not Greater Than Or Equal | \$1{\$2=%NOT >= \$2} |
| \$1: CMP_NOT_GE__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_NOT_GE__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-----------------------|---------------|---------------------|
| CMP_NOT_LT | Not Less Than | \$1{\$2=%NOT < \$2} |
| \$1: CMP_NOT_LT__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_NOT_LT__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-----------------------|------------------------|----------------------|
| CMP_NOT_LE | Not Less Than Or Equal | \$1{\$2=%NOT <= \$2} |
| \$1: CMP_NOT_LE__ARG1 | Arg1 | rel-arith-expr |
| \$2: CMP_NOT_LE__ARG2 | Arg2 | rel-arith-expr |

| | | |
|-------------------------|-------------|--------------------|
| CMP_NOT_NUM | Not Numeric | \$1 [NOT NUMERIC] |
| \$1: CMP_NOT_NUM__IDENT | Identifier | identifier |

| | | |
|--------------------------|----------------|-----------------------|
| CMP_NOT_ALPH | Not Alphabetic | \$1 [NOT ALPHABETIC] |
| \$1: CMP_NOT_ALPH__IDENT | Identifier | identifier |

| | | |
|-----------------------------------|----------------------|-----------------------------|
| CMP_NOT_ALPH_LOWER | Not Alphabetic-Lower | \$1 [NOT ALPHABETIC-LOWER] |
| \$1: CMP_NOT_ALPH_LOWER__IDENT | Identifier | identifier |

| | | |
|-----------------------------------|----------------------|-----------------------------|
| CMP_NOT_ALPH_UPPER | Not Alphabetic-Upper | \$1 [NOT ALPHABETIC-UPPER] |
| \$1: CMP_NOT_ALPH_UPPER__IDENT | Identifier | identifier |

| | | |
|-------------------------|--------------|--------------------|
| CMP_NOT_POS | Not Positive | \$1 [NOT POSITIVE] |
| \$1: CMP_NOT_POS__IDENT | Identifier | identifier |

| | | |
|-------------------------|--------------|--------------------|
| CMP_NOT_NEG | Not Negative | \$1 [NOT NEGATIVE] |
| \$1: CMP_NOT_NEG__IDENT | Identifier | identifier |

| | | |
|-------------------------|------------|----------------|
| CMP_NOT_ZER | Not Zero | \$1 [NOT ZERO] |
| \$1: CMP_NOT_ZER__IDENT | Identifier | identifier |

| | | |
|---------------------------------|--------------|----------------------|
| CMP_IS_NOT_CLASS | Not Of Class | \$1{\$2= IS NOT \$2} |
| \$1: CMP_IS_NOT_CLASS__IDENT | Identifier | identifier |
| \$2: CMP_IS_NOT_CLASS__CLASS | Identifier | NAMED_OBJ_USE |

| | | |
|---------|-----|--------|
| LOG_AND | And | @%AND& |
|---------|-----|--------|

| | | |
|--------|----|-------|
| LOG_OR | Or | @%OR& |
|--------|----|-------|

10.8 Wang Extension

10.8.1 Figurative Constants

| | | |
|--------------------|--------------------------------|---|
| PARA_FIG_CONSTS | Figurative Constants Paragraph | FIGURATIVE-CONSTANTS.\n\t\$1.\b\n{ \$2? |
| \$1: FIGCONST_LIST | Fig Constant Definiton List | EX_FIG_CONST_DEF_LIST (EX_FIG_CONST_DEF) |
| \$2: FIGCONST_COPY | Copy Begin/End | copy-expr |

| | | |
|-----------------------|-----------------------------|-----|
| EX_FIG_CONST_DEF_LIST | Fig Constant Definiton List | @\n |
|-----------------------|-----------------------------|-----|

| | | |
|----------------------|------------------------|----------------------|
| EX_FIG_CONST_DEF | Fig Constant Definiton | \$1\$p4{\$2= IS \$2} |
| \$1: FIGCONDEF_DEFD | Constant Name Defined | NAMED_OBJ_DEF |
| \$2: FIGCONDEF_VALUE | Constant Value | STR_CONST |

10.8.2 File Control Entry

| | | |
|--------------------------|--------------------|---|
| FCX_ASSIGN_WANG | Assign Clause Wang | ASSIGN TO \$1{\$2= \$2}{\$3? \$3}{\$4? \$4} |
| \$1: ASSTOWANG_FILE | File Reference | NAMED_OBJ_USE STR_CONST |
| \$2: ASSTOWANG_DEVICE | Device Type | STR_CONST |
| \$3: ASSTOWANG_DISPLAY | Display Phrase | FCX_NODISPLAY FCX_DISPLAY NULL |
| \$4: ASSTOWANG_RESPECIFY | Respecify Phrase | FCX_NORESPECIFY FCX_RESPECIFY NULL |

| | | |
|---------------|------------------|-----------|
| FCX_NODISPLAY | Nodisplay Phrase | NODISPLAY |
|---------------|------------------|-----------|

| | | |
|-------------|----------------|---------|
| FCX_DISPLAY | Display Phrase | DISPLAY |
|-------------|----------------|---------|

| | | |
|-----------------|--------------------|-------------|
| FCX_NORESPECIFY | Norespecify Phrase | NORESPECIFY |
|-----------------|--------------------|-------------|

| | | |
|---------------|------------------|-----------|
| FCX_RESPECIFY | Respecify Phrase | RESPECIFY |
|---------------|------------------|-----------|

| | | |
|--------------------------------------|---------------------------------|---|
| FCX_WANG_ALT_REC_KEY | Alternate Record Key Wang | ALTERNATE RECORD KEY \$1 |
| \$1: FCX_WANG_ALT_REC_KEY__ALTREC | Key Components KEYWANG_COMPS | FCX_WANG_KEY_COMP_LIST (FCX_WANG_KEY_COMP) |

| | | |
|------------------------|--------------------------------|----|
| FCX_WANG_KEY_COMP_LIST | Wang Record Key Component List | @& |
|------------------------|--------------------------------|----|

| | | |
|---------------------------|-------------------------------|--|
| FCX_WANG_KEY_COMP | Alt Record Key Wang Component | { \$1? \$1 IS } { \$2= \$2 } { \$3? WITH DUPLICATES } |
| \$1: ARKWANGCOMP_PRTY | Priority | INT_NUM_CONST NULL |
| \$2: ARKWANGCOMP_DATAITEM | Key Data Item | NAMED_OBJ_USE |
| \$3: ARKWANGCOMP_DUPLS | With Duplicates | FCX_ALTKEY_WITH_DUPLICATES NULL |

| | | |
|--|------------------------|-------------------------|
| FCX_CURSOR_POSN | Cursor Position Clause | CURSOR POSITION IS \$1 |
| \$1: FCX_CURSOR_POSN__WANGCURPOS_DATAITEM | Data Item | NAMED_OBJ_USE |

| | | |
|--|--------------------|---------------------|
| FCX_BUFFER_SIZE | Buffer Size Clause | BUFFER SIZE IS \$1 |
| \$1: FCX_BUFFER_SIZE__WANGBUFSIZE_VALUE | Value | INT_NUM_CONST |

| | | |
|---------------------------------------|--------------|---------------|
| FCX_PFKEY | Pfkey Clause | PFKEY IS \$1 |
| \$1: FCX_PFKEY__WANGPFKEY_DATAITEM | Data Item | NAMED_OBJ_USE |

10.8.3 File Description Entry

| | | |
|----------------|-------------------|------------|
| FDX_COMPRESSED | Compressed Phrase | COMPRESSED |
|----------------|-------------------|------------|

| | | |
|-----------------|-----------------|----------|
| FDX_VO_FILENAME | Filename Phrase | FILENAME |
|-----------------|-----------------|----------|

| | | |
|----------------|----------------|---------|
| FDX_VO_LIBRARY | Library Phrase | LIBRARY |
|----------------|----------------|---------|

| | | |
|---------------|---------------|--------|
| FDX_VO_VOLUME | Volume Phrase | VOLUME |
|---------------|---------------|--------|

| | | |
|--------------|--------------|-------|
| FDX_VO_SPACE | Space Phrase | SPACE |
|--------------|--------------|-------|

| | | |
|-----------------|-----------------|----------|
| FDX_VO_POSITION | Position Phrase | POSITION |
|-----------------|-----------------|----------|

| | | |
|-------------------|-------------------|------------|
| FDX_VO_INDEX_AREA | Index Area Phrase | INDEX AREA |
|-------------------|-------------------|------------|

| | | |
|------------------|------------------|-----------|
| FDX_VO_DATA_AREA | Data Area Phrase | DATA AREA |
|------------------|------------------|-----------|

| | | |
|--------------------|--------------------|-------------|
| FDX_VO_PRINT_CLASS | Print-Class Phrase | PRINT-CLASS |
|--------------------|--------------------|-------------|

| | | |
|-------------------|-------------------|------------|
| FDX_VO_FILE_CLASS | File-Class Phrase | FILE-CLASS |
|-------------------|-------------------|------------|

| | | |
|--------------------|--------------------|-------------|
| FDX_VO_EXTENT_SIZE | Extent Size Phrase | EXTENT SIZE |
|--------------------|--------------------|-------------|

| | | |
|------------------------|------------------------|-----------------|
| FDX_VO_RECOVERY_BLOCKS | Recovery-Blocks Phrase | RECOVERY-BLOCKS |
|------------------------|------------------------|-----------------|

| | | |
|------------------------|------------------------|-----------------|
| FDX_VO_RECOVERY_STATUS | Recovery-Status Phrase | RECOVERY-STATUS |
|------------------------|------------------------|-----------------|

| | | |
|----------------------|----------------------|---------------|
| FDX_VO_DATABASE_NAME | Database-Name Phrase | DATABASE-NAME |
|----------------------|----------------------|---------------|

10.8.4 Record Description Entry

| | | |
|--|----------------------------|--|
| DECL_DD_ENTRY_WANG_DISPLAY | Wang Data Item Declaraiton | \r\$1{\$2? \$2}{\$3? \$3}{\$4? 4}{\$4}{\$6? 6} |
| \$1: DECL_DD_ENTRY_WANG_DISPLAY__DD_LEVEL | Level Number | INT_NUM_CONST |
| \$2: DECL_DD_ENTRY_WANG_DISPLAY__DD_COPY_BEf_NAME | Copy before name | copy-expr NULL |
| \$3: DECL_DD_ENTRY_WANG_DISPLAY__DD_DATA_NAME | Data Item Defined | NAMED_OBJ_DEF EX_FILLER NULL |
| \$4: DECL_DD_ENTRY_WANG_DISPLAY__DD_REDEFINES | Redefines | DX_REDEFINES NULL |
| \$5: DECL_DD_ENTRY_WANG_DISPLAY__DD_PICTURE | Picture | DX_PICTURE NULL |
| \$6: DECL_DD_ENTRY_WANG_DISPLAY__DD_USAGE | Usage | DX_USAGE NULL |
| \$7: DECL_DD_ENTRY_WANG_DISPLAY__DD_BWZ | Blank When Zero | DX_BWZ NULL |
| \$8: DECL_DD_ENTRY_WANG_DISPLAY__DD_JUSTIFIED | Justified | DX_JUSTIFIED NULL |
| \$9: DECL_DD_ENTRY_WANG_DISPLAY__DD_SIGN | Sign | DX_SIGN NULL |
| \$10: DECL_DD_ENTRY_WANG_DISPLAY__DD_SYNCHRONIZED | Synchronized | DX_SYNCHRONIZED NULL |
| \$11: DECL_DD_ENTRY_WANG_DISPLAY__DD_EXTERNAL | External | DX_EXTERNAL NULL |
| \$12: DECL_DD_ENTRY_WANG_DISPLAY__DD_GLOBAL | Global | DX_GLOBAL NULL |
| \$13: DECL_DD_ENTRY_WANG_DISPLAY__DD_RENAMES | Renames | DX_RENAMES NULL |
| \$14: DECL_DD_ENTRY_WANG_DISPLAY__DD_OCCURS | Occurs | DX_OCCURS NULL |
| \$15: DECL_DD_ENTRY_WANG_DISPLAY__DD_VALUE | Value | DX_VALUE NULL |
| \$16: DECL_DD_ENTRY_WANG_DISPLAY__DD_UNINLINED_COPY | Copy Begin/End | copy-expr NULL |
| \$17: DECL_DD_ENTRY_WANG_DISPLAY__DD_CATEGORY | Category | DCAT_GROUP DCAT_INDEX DCAT_POINTER DCAT_PROC_POINTER PIC_ALPHA PIC_NUM PIC_ALPHA_NUM |

| | | |
|--------------------------------------|---------------|---------------|
| EX_WDD_COLUMN | Column Clause | COLUMN \$1 |
| \$1: EX_WDD_COLUMN__WDD_COL_VALUE | Value | INT_NUM_CONST |

| | | |
|-----------------------------------|------------|---------------|
| EX_WDD_ROW | Row Clause | ROW \$1 |
| \$1: EX_WDD_ROW__WDD_ROW_VALUE | Value | INT_NUM_CONST |

| | | |
|---|---------------|---------------|
| EX_WDD_OBJECT | Object Clause | OBJECT IS \$1 |
| \$1: EX_WDD_OBJECT__WDD_OBJ_DATAITEM | Data Item | NAMED_OBJ_USE |

| | | |
|---|---------------|---------------|
| EX_WDD_SOURCE | Source Clause | SOURCE IS \$1 |
| \$1: EX_WDD_SOURCE__WDD_SRC_DATAITEM | Data Item | NAMED_OBJ_USE |

| | | |
|--------------------------------------|--------------|---|
| EX_WDD_RANGE | Range Clause | RANGE IS \$1 |
| \$1: EX_WDD_RANGE__WDD_RANGE_TYPE | Range Type | EX_WDD_NEGATIVE EX_WDD_POSITIVE EX_WDD_FROM EX_WDD_TABLE |

| | | |
|-----------------|----------------|----------|
| EX_WDD_NEGATIVE | Negative Range | NEGATIVE |
|-----------------|----------------|----------|

| | | |
|-----------------|----------------|----------|
| EX_WDD_POSITIVE | Positive Range | POSITIVE |
|-----------------|----------------|----------|

| | | |
|--------------------|------------|----------------------------|
| EX_WDD_FROM | From Range | FROM \$1{\$2= TO \$2} |
| \$1: WDD_FROM_FROM | From | NAMED_OBJ_USE literal |
| \$2: WDD_FROM_TO | From | NAMED_OBJ_USE literal |

| | | |
|------------------------------------|-------------|---------------|
| EX_WDD_TABLE | Table Range | \$1 |
| \$1: EX_WDD_TABLE__WDD_TABLE_ID | Table | NAMED_OBJ_USE |

| | | |
|---------------|------------------|------------|
| DU_DISPLAY_WS | Display-WS Usage | DISPLAY-WS |
|---------------|------------------|------------|

10.8.5 Statements

| | | |
|------------------------|-----------------|--|
| STMT_BEGIN | Begin Statement | BEGIN \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: WBEGIN_TRANS | Trans/SubTrans | EX_WANG_TRANSACTION EX_WANG_SUBTRANSACTION NULL |
| \$2: WBEGIN_ONERROR | On Error | EX_ON_ERROR NULL |
| \$3: WBEGIN_NOTONERROR | Not On Error | EX_NOT_ON_ERROR NULL |
| \$4: WBEGIN_END_BEGIN | End-Begin | EX_END_BEGIN NULL |

| | | |
|---------------------|--------------------|-------------|
| EX_WANG_TRANSACTION | Transaction Phrase | TRANSACTION |
|---------------------|--------------------|-------------|

| | | |
|------------------------|-----------------------|----------------|
| EX_WANG_SUBTRANSACTION | Subtransaction Phrase | SUBTRANSACTION |
|------------------------|-----------------------|----------------|

| | | |
|--------------|-----------|-----------|
| EX_END_BEGIN | End-Begin | END-BEGIN |
|--------------|-----------|-----------|

| | | |
|--------------------------|-----------------------|---|
| STMT_DISPLAY_AND_READ | Display And Read stmt | DISPLAY AND READ{\$1? \$1}{\$2= \$2}{\$3= |
| \$1: WDSPLRD_ALTERED | Altered | EX_DNR_ALTERED NULL |
| \$2: WDSPLRD_WHAT | What | NAMED_OBJ_USE |
| \$3: WDSPLRD_ONWHAT | On What | NAMED_OBJ_USE |
| \$4: WDSPLRD_PFKEYS | Pfkeys | EX_DNR_ONLY_PFKEYS NULL |
| \$5: WDSPLRD_NOMOD | No-Mod | EX_DNR_NO_MOD NULL |
| \$6: WDSPLRD_END_DISPLAY | End-Display | EX_END_DISPLAY NULL |

| | | |
|----------------|----------------|---------|
| EX_DNR_ALTERED | Altered Phrase | ALTERED |
|----------------|----------------|---------|

| | | |
|---------------------|--------------------|----------------------------|
| EX_DNR_ONLY_PFKEYS | Only Pfkeys Phrase | ONLY PFKEYS \$1{\$2?\n\$2} |
| \$1: WDNR_ONLY_KEYS | Values | EX_LIST (ident-liter) |
| \$2: WDNR_ONLY_ON | Values | EX_DNR_ON NULL |

| | | |
|--------------------|------------------|------------------------------|
| EX_DNR_ON | On Pfkeys Phrase | ON PFKEYS \$1{\$2?\n\t\$2\b} |
| \$1: DNR_ON_PFKEYS | Pfkeys | EX_LIST (ident-liter) |
| \$2: DNR_ON_STMTS | Statements | STMT_LIST (statement) |

| | | |
|--|---------------|-----------------------|
| EX_DNR_NO_MOD | No Mod Phrase | NO-MOD\n\t\$1\b |
| \$1: EX_DNR_NO_MOD__DNR_NOMOD_STMTS | Statements | STMT_LIST (statement) |

| | | |
|-----------------------|---------------|---|
| STMT_FREE_ALL | Free All stmt | FREE ALL{\$1?\n\$1}{\$2?\n\$2}{\$3?\n\$3} |
| \$1: WFREE_ONERROR | On Error | EX_ON_ERROR NULL |
| \$2: WFREE_NOTONERROR | Not On Error | EX_NOT_ON_ERROR NULL |
| \$3: WFREE_END_FREE | End-Begin | EX_END_FREE NULL |

| | | |
|-------------|----------|----------|
| EX_END_FREE | End Free | END-FREE |
|-------------|----------|----------|

| | | |
|---------------------|-------------|--|
| STMT_HOLD | Hold stmt | HOLD{\$1? \$1}{\$2= \$2}{\$3? \$3}{\$4? \$4} |
| \$1: WHOLD_LIST | List Phrase | EX_HOLD_LIST NULL |
| \$2: WHOLD_RECORDS | Records | EX_HOLD_RECORDS NULL |
| \$3: WHOLD_TIMEOUT | Timeout | EX_WANG_TIMEOUT NULL |
| \$4: WHOLD_END_HOLD | End-Hold | EX_END_HOLD NULL |

| | | |
|--------------|------------------|------|
| EX_HOLD_LIST | Hold List Phrase | LIST |
|--------------|------------------|------|

| | | |
|---------------------|---------------------|---|
| EX_HOLD_RECORDS | Hold Records Phrase | RECORDS OF \$1{\$2= FOR \$2}{\$3? \$3} |
| \$1: WHOLDRECS_FILE | File | NAMED_OBJ_USE |
| \$2: WHOLDRECS_TYPE | Retrieval/Update | EX_RETRIEVAL EX_UPDATE |
| \$3: WHOLDRECS_KEYS | With Keys | EX_WITH_KEYS NULL |

| | | |
|-------------------------------------|------------------|--------------------------------|
| EX_WITH_KEYS | With Keys Phrase | WITH KEYS \$1 |
| \$1: EX_WITH_KEYS__WITHKEYS_KEYS | File | EX_HOLD_KEY_LIST (EX_HOLD_KEY) |

| | | |
|--------------|------------------|-----------|
| EX_RETRIEVAL | Retrieval Phrase | RETRIEVAL |
|--------------|------------------|-----------|

| | | |
|-----------|---------------|--------|
| EX_UPDATE | Update Phrase | UPDATE |
|-----------|---------------|--------|

| | | |
|------------------|--------------|---|
| EX_HOLD_KEY_LIST | Hold KeyList | @ |
|------------------|--------------|---|

| | | |
|-----------------------|---------------|----------------------------|
| EX_HOLD_KEY | Hold Key | {\$1? \$1 }{\$2= \$2} |
| \$1: WHOLDKEY_INITIAL | Initial | EX_HOLD_INITIAL NULL |
| \$2: WHOLDKEY_CHARSOF | Characters Of | NAMED_OBJ_USE literal |

| | | |
|---------------------------------------|----------------|----------------------------|
| EX_HOLD_INITIAL | Initial Phrase | INITIAL \$1 |
| \$1: EX_HOLD_INITIAL__HOLD_INITIAL | File L.NAME | NAMED_OBJ_USE literal |

| | | |
|------------------------|----------------|--|
| EX_WANG_TIMEOUT | Timeout Phrase | TIMEOUT OF \$1 SECONDS{\$2? \$2}{\$3? \n\ |
| \$1: WTIMEOUT_SECONDS | Seconds | NAMED_OBJ_USE literal |
| \$2: WTIMEOUT HOLDERID | Holder ID | EX HOLDER_ID NULL |
| \$3: WTIMEOUT_STMT | Statement | statement |

| | | |
|--------------------------------------|------------------|---------------|
| EX_HOLDER_ID | Holder-Id Phrase | HOLDER-ID \$1 |
| \$1: EX_HOLDER_ID__HOLDER_ID_NAME | File | NAMED_OBJ_USE |

| | | |
|-------------|----------|----------|
| EX_END_HOLD | End-Hold | END-HOLD |
|-------------|----------|----------|

| | | |
|-------------------------|---------------------------|--|
| STMT_MOVE_WITH_CONV | Move With Conversion stmt | MOVE WITH CONVERSION \$1{\$2= TO \$2}{\$3= FROM \$3} |
| \$1: WMOVECONV_SOURCE | Source | identifier |
| \$2: WMOVECONV_DEST | Destination | identifier |
| \$3: WMOVECONV_ONERROR | On Error | EX_ON_ERROR NULL |
| \$4: WMOVECONV_END_MOVE | End-Move | EX_END_MOVE NULL |

| | | |
|-------------|-----------------|----------|
| EX_END_MOVE | End-Move phrase | END-MOVE |
|-------------|-----------------|----------|

| | | |
|------------------|----------------------|------------------|
| EX_BIT_OF | BitOf | \$1 OF{\$2= \$2} |
| \$1: WBITOF_MASK | Bit Mask (Fig Const) | NAMED_OBJ_USE |
| \$2: WBITOF_WHAT | Bit Of What | NAMED_OBJ_USE |

| | | |
|-----------------------------|--------------------|---------------|
| EX_FAC_OF | Move-Fac Of Phrase | FAC OF \$1 |
| \$1: EX_FAC_OF__WFACOF_WHAT | Fac Of What | NAMED_OBJ_USE |

| | | |
|--|---------------------------|-------------------|
| EX_ORDER_AREA_OF | Move-Order Area Of Phrase | ORDER-AREA OF \$1 |
| \$1: EX_ORDER_AREA_OF__WORDERAREOF_WHAT | Order Area Of What | NAMED_OBJ_USE |

| | | |
|-------------|---------------|--------|
| IOCX_SHARED | Shared Phrase | SHARED |
|-------------|---------------|--------|

| | | |
|--------------------|----------------|---------------|
| IOCX_SPECIAL_INPUT | Special Phrase | SPECIAL-INPUT |
|--------------------|----------------|---------------|

| | | |
|-------------------|------------------|-----------|
| EX_READ_WITH_HOLD | With Hold Phrase | WITH HOLD |
|-------------------|------------------|-----------|

| | | |
|--------------------|-------------------|------------|
| EX_READ_MODIFIABLE | Modifiable Phrase | MODIFIABLE |
|--------------------|-------------------|------------|

| | | |
|-----------------|----------------|---------|
| EX_READ_ALTERED | Altered Phrase | ALTERED |
|-----------------|----------------|---------|

| | | |
|----------------|--------------------|-------------|
| EX_AFTER_ALARM | After Alarm Phrase | AFTER ALARM |
|----------------|--------------------|-------------|

| | | |
|--|-----------------------------|---|
| EX_AFTER_SET_CURSOR | After Setting Cursor phrase | AFTER SETTING CURSOR COLUMN \$1{\$2= R |
| \$1: WAFTERSETCURSOR.COLUMN | Column | NAMED_OBJ_USE literal |
| \$2: WAFTERSETCURSOR.ROW | Row | NAMED_OBJ_USE literal |
| EX_AFTER_ROLL_DOWN | After Roll Down Phrase | ROLL DOWN |
| EX_AFTER_ROLL_UP | After Roll Up Phrase | ROLL UP |
| EX_AFTER_ERASE_PROTECT | After Erase Protect Phrase | ERASE PROTECT |
| EX_AFTER_ERASE_MODIFY | After Erase Modify Phrase | ERASE MODIFY |
| EX_ON_ERROR | On Error stmt | ON ERROR\n\t\$1\b |
| \$1: EX_ON_ERROR__ONERROR_STMTS | Statements | STMT_LIST (statement) |
| EX_NOT_ON_ERROR | Not On Error stmt | NOT ON ERROR\n\t\$1\b |
| \$1: EX_NOT_ON_ERROR__NOTONERROR_STMTS | Statements | STMT_LIST (statement) |
| STMT_USE_AFTER_DEADLOCK | Use After Deadlock stmt | USE AFTER DEADLOCK |
| STMT_ROLLBACK_WANG | Rollback Statement | ROLLBACK \$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$ |
| \$1: WROLLBACK_FILE | File | NAMED_OBJ_USE |
| \$2: WROLLBACK_ONERROR | On Error | EX_ON_ERROR NULL |
| \$3: WROLLBACK_NOTONERROR | Not On Error | EX_NOT_ON_ERROR NULL |
| \$4: WROLLBACK_END_ROLLBACK | End-Rollback | EX_END_ROLLBACK NULL |
| EX_END_ROLLBACK | End-Rollback Phrase | END-ROLLBACK |
| START_KEY_WANG | Start Key Phrase | KEY \$1 |
| \$1: START_KEY_WANG__START_KEY_WANG_IDENT | Key Ident | NAMED_OBJ_USE |

| | | |
|---------------------------------------|----------------|-------------|
| CMP_ALTERED | Altered Phrase | \$1 ALTERED |
| \$1: CMP_ALTERED__CMP_ALTERED_WHAT | What | arith-expr |

10.9 SQL Extension

10.9.1 Top Level Statements

| | | |
|------------------------------------|--------------------------------|--|
| STMT_EXEC_SQL_IN_DCL | Declarative EXEC SQL statement | \tEXEC SQL\b\n\t\t\$1\b\b\n\tEND-EXEC |
| \$1: STMT_EXEC_SQL_IN_DCL__STMT | Statement | SQL_STMT_BEGIN_DECLARE_SECTION SQL_STMT_END_DECLARE_SECTION |

| | | |
|---|-----------------------|----------------------------------|
| STMT_UNINLINED_INCLUDE | UnInlined Sql Include | EXEC SQL INCLUDE \$1 END-EXEC[.] |
| \$1: STMT_UNINLINED_INCLUDE__FILE | File Name | CHAR_STRING |
| \$2: STMT_UNINLINED_INCLUDE__UNUSED_REPLNG | Unused | NULL |
| \$3: STMT_UNINLINED_INCLUDE__REAL_FILE | Real File Name | CHAR_STRING |
| \$4: STMT_UNINLINED_INCLUDE__LEVEL | Nesting Level | INT_NUM_CONST |

| | | |
|---|-----------------------|----------------------------------|
| STMT_INLINED_INCLUDE_BEG | SQL Include Statement | EXEC SQL INCLUDE \$1 END-EXEC[.] |
| \$1: STMT_INLINED_INCLUDE_BEG__FILE | File To Include | CHAR_STRING |
| \$2: STMT_INLINED_INCLUDE_BEG__UNUSED_REPLNG | Ununsed | CHAR_STRING |
| \$3: STMT_INLINED_INCLUDE_BEG__REAL_FILE | Real File Name | CHAR_STRING |
| \$4: STMT_INLINED_INCLUDE_BEG__LEVEL | Nesting Level | INT_NUM_CONST |

| | | |
|---|------------------|---------------|
| STMT_INLINED_INCLUDE_END | Inlined Copy End | |
| \$1: STMT_INLINED_INCLUDE_END__REAL_FILE | Real File Name | CHAR_STRING |
| \$2: STMT_INLINED_INCLUDE_END__LEVEL | Nesting Level | INT_NUM_CONST |

| | | |
|-------------------------|-------------------------------|---|
| STMT_EXEC_SQL | Imperative EXEC SQL statement | EXEC SQL\n\t\$1\b\nEND-EXEC |
| \$1: STMT_EXEC_SQL_STMT | Statement | SQL_STMT_WHENEVER SQL_STMT_DECLARE_TABLE SQL_STMT_CONNECT SQL_STMT_DISCONNECT SQL_STMT_SET_CONNECTION SQL_STMT_SET_CATALOG SQL_STMT_SET_SCHEMA SQL_STMT_SET_NAMES SQL_STMT_SET_SESSION_AUTH SQL_STMT_COMMIT SQL_STMT_ROLLBACK SQL_STMT_SELECT SQL_STMT_INSERT SQL_STMT_DELETE SQL_STMT_UPDATE SQL_STMT_DECLARE_CURSOR SQL_STMT_OPEN_CURSOR SQL_STMT_CLOSE_CURSOR SQL_STMT_FETCH_CURSOR SQL_STMT_DELETE_CURSOR SQL_STMT_UPDATE_CURSOR SQL_STMT_ALLOCATE_DESCRIPTOR SQL_STMT_ALLOCATE_CURSOR SQL_STMT_DEALLOCATE_DESCRIPTOR SQL_STMT_DEALLOCATE_PREPARE SQL_STMT_DESCRIBE_INPUT SQL_STMT_DESCRIBE_OUTPUT SQL_STMT_EXECUTE_IMMEDIATE SQL_STMT_EXECUTE_PREPARED SQL_STMT_PREPARE SQL_STMT_GET_DESCRIPTOR SQL_STMT_SET_DESCRIPTOR |

10.9.2 Declarative Statements

| | | |
|--------------------------------|-----------------------|-----------------------|
| SQL_STMT_BEGIN_DECLARE_SECTION | Begin Declare Section | BEGIN DECLARE SECTION |
| SQL_STMT_END_DECLARE_SECTION | End Declare Section | END DECLARE SECTION |

10.9.3 WHENEVER Statement

| | | |
|----------------------------|--------------------|---|
| SQL_STMT_WHENEVER | Whenever Statement | WHENEVER \$1{\$2=\$2} |
| \$1: SQL_STMT_WHENEVER.ERR | Error Type | SQL_ERR_SQLERROR SQL_ERR_NOT_FOUND |
| \$2: SQL_STMT_WHENEVER.ACT | Action | SQL_ACT_GOTO SQL_ACT_CONTINUE |

| | | |
|------------------|---------------|----------|
| SQL_ERR_SQLERROR | Sqerror Error | SQLERROR |
|------------------|---------------|----------|

| | | |
|-------------------|-----------------|-----------|
| SQL_ERR_NOT_FOUND | Not Found Error | NOT FOUND |
|-------------------|-----------------|-----------|

| | | |
|--|--------------|---------------|
| SQL_ACT_GOTO | Go To Action | GO TO \$1 |
| \$1: SQL_ACT_GOTO__SQL_ACT_GOTO_LABEL | Label | NAMED_OBJ_USE |

| | | |
|------------------|-----------------|----------|
| SQL_ACT_CONTINUE | Continue Action | CONTINUE |
|------------------|-----------------|----------|

| | | |
|---|-------------------------|------------------------------------|
| SQL_STMT_DECLARE_TABLE | Declare Table Statement | DECLARE \$1 TABLE\n\t({\$2=\$2})\b |
| \$1: SQL_STMT_DECLARE_TABLE_TABLE | Table Name | SQL_TABLE_REF |
| \$2: SQL_STMT_DECLARE_TABLE_COL_DEFS | Column Definition List | SQL_COL_DEF_LIST (SQL_COL_DEF) |

| | | |
|------------------|------------------------|-----|
| SQL_COL_DEF_LIST | Column Definition List | @,& |
|------------------|------------------------|-----|

| | | |
|--------------------------------|-------------------|---|
| SQL_COL_DEF | Column Definition | \$1{\$2= \$2}{\$3? \$3} |
| \$1: SQL_COL_DEF_NAME | Name | CHAR_STRING |
| \$2: SQL_COL_DEF_TYPE | Type | SQL_TYPE_CHAR SQL_TYPE_VARCHAR SQL_TYPE_BIT SQL_TYPE_VARBIT SQL_TYPE_NUMERIC SQL_TYPE_DECIMAL SQL_TYPE_INT SQL_TYPE_SMALLINT SQL_TYPE_FLOAT SQL_TYPE_REAL SQL_TYPE_DOUBLE_PRECISION |
| \$3: SQL_COL_DEF_CONSTRAINT | Constraint | SQL_CONSTR_NOT_NULL SQL_CONSTR_PRIMARY_KEY SQL_CONSTR_UNIQUE SQL_CONSTR_CHECK |

| | | |
|---------------------------------|-----------------------|-------------------------|
| SQL_TYPE_CHAR | Char Type | CHAR (\$1) |
| \$1: SQL_TYPE_CHAR_LENGTH | Length | INT_NUM_CONST |
| SQL_TYPE_VARCHAR | VarChar Type | VARCHAR (\$1) |
| \$1: SQL_TYPE_VARCHAR_LENGTH | Length | INT_NUM_CONST |
| SQL_TYPE_BIT | Bit Type | BIT (\$1) |
| \$1: SQL_TYPE_BIT_LENGTH | Length | INT_NUM_CONST |
| SQL_TYPE_VARBIT | VarBit Type | VARBIT (\$1) |
| \$1: SQL_TYPE_VARBIT_LENGTH | Length | INT_NUM_CONST |
| SQL_TYPE_NUMERIC | Numeric Type | NUMERIC (\$1{\$2?,\$2}) |
| \$1: SQL_TYPE_NUMERIC_DIGITS | Digits | INT_NUM_CONST |
| \$2: SQL_TYPE_NUMERIC_DOTPOS | Dot Position | INT_NUM_CONST NULL |
| SQL_TYPE_DECIMAL | Decimal Type | DECIMAL (\$1{\$2?,\$2}) |
| \$1: SQL_TYPE_DECIMAL_DIGITS | Digits | INT_NUM_CONST |
| \$2: SQL_TYPE_DECIMAL_DOTPOS | Dot Position | INT_NUM_CONST NULL |
| SQL_TYPE_INT | Integer Type | INT |
| SQL_TYPE_SMALLINT | Small Integer Type | SMALLINT |
| SQL_TYPE_FLOAT | Float Type | FLOAT{\$1? (\$1)} |
| \$1: SQL_TYPE_FLOAT_DIGITS | Digits | INT_NUM_CONST NULL |
| SQL_TYPE_REAL | Real Type | REAL |
| SQL_TYPE_DOUBLE_PRECISION | Double Precision Type | DOUBLE PRECISION |

| | | |
|---------------|-----------|------|
| SQL_TYPE_DATE | Date Type | DATE |
|---------------|-----------|------|

| | | |
|---------------|-----------|------|
| SQL_TYPE_TIME | Time Type | TIME |
|---------------|-----------|------|

| | | |
|--------------------|----------------|-----------|
| SQL_TYPE_TIMESTAMP | Timestamp Type | TIMESTAMP |
|--------------------|----------------|-----------|

| | | |
|---------------------|---------------------|----------|
| SQL_CONSTR_NOT_NULL | Not Null Constraint | NOT NULL |
|---------------------|---------------------|----------|

| | | |
|------------------------|------------------------|-------------|
| SQL_CONSTR_PRIMARY_KEY | Primary Key Constraint | PRIMARY KEY |
|------------------------|------------------------|-------------|

| | | |
|-------------------|-------------------|--------|
| SQL_CONSTR_UNIQUE | Unique Constraint | UNIQUE |
|-------------------|-------------------|--------|

| | | |
|-----------------------------|------------------|-------------|
| SQL_CONSTR_CHECK | Check Constraint | CHECK (\$1) |
| \$1: SQL_CONSTR_CHECK__COND | Condition | sql-cond |

10.9.4 Connect and Transaction Statements

| | | |
|----------------------------------|-------------------|---|
| SQL_STMT_CONNECT | Connect Statement | CONNECT TO \$1 |
| \$1: SQL_STMT_CONNECT__SERVER | Server | SQL_SERVER_DEFAULT SQL_SERVER_SPEC |

| | | |
|-------------------------------------|----------------------|---|
| SQL_STMT_DISCONNECT | Disconnect Statement | DISCONNECT TO \$1 |
| \$1: SQL_STMT_DISCONNECT__SERVER | Server | SQL_SERVER_DEFAULT SQL_SERVER_CURRENT SQL_SERVER_ALL SQL_SERVER_SPEC |

| | | |
|---|--------------------------|---|
| SQL_STMT_SET_CONNECTION | Set Connection Statement | SET CONNECTION \$1 |
| \$1: SQL_STMT_SET_CONNECTION__SERVER | Server | SQL_SERVER_DEFAULT SQL_SERVER_SPEC |

| | | |
|------------------------------------|-----------------------|---------------------------------|
| SQL_STMT_SET_CATALOG | Set Catalog Statement | SET CATALOG \$1 |
| \$1: SQL_STMT_SET_CATALOG__NAME | Name | STR_CONST SQL_HOST_VAR_REF |

| | | |
|-----------------------------------|----------------------|---------------------------------|
| SQL_STMT_SET_SCHEMA | Set SCHEMA Statement | SET SCHEMA \$1 |
| \$1: SQL_STMT_SET_SCHEMA__NAME | Name | STR_CONST SQL_HOST_VAR_REF |

| | | |
|----------------------------------|---------------------|---------------------------------|
| SQL_STMT_SET_NAMES | Set Names Statement | SET NAMES \$1 |
| \$1: SQL_STMT_SET_NAMES__NAME | Name | STR_CONST SQL_HOST_VAR_REF |

| | | |
|---|----------------------------|---------------------------------|
| SQL_STMT_SET_SESSION_AUTH | Set Session Auth Statement | SET SESSION AUTHORIZATION \$1 |
| \$1: SQL_STMT_SET_SESSION_AUTH__NAME | Name | STR_CONST SQL_HOST_VAR_REF |

| | | |
|-----------------|------------------|--------|
| SQL_STMT_COMMIT | Commit Statement | COMMIT |
|-----------------|------------------|--------|

| | | |
|-------------------|--------------------|----------|
| SQL_STMT_ROLLBACK | Rollback Statement | ROLLBACK |
|-------------------|--------------------|----------|

| | | |
|--------------------|----------------|---------|
| SQL_SERVER_DEFAULT | Default Server | DEFAULT |
|--------------------|----------------|---------|

| | | |
|--------------------|----------------|---------|
| SQL_SERVER_CURRENT | Current Server | CURRENT |
|--------------------|----------------|---------|

| | | |
|----------------|-------------|-----|
| SQL_SERVER_ALL | All Servers | ALL |
|----------------|-------------|-----|

| | | |
|---------------------------|----------------------|---------------------------------|
| SQL_SERVER_SPEC | Server Specification | \$1{\$2? \$2}{\$3? \$3} |
| \$1: SQL_SERVER_SPEC_NAME | Server | STR_CONST SQL_HOST_VAR_REF |
| \$2: SQL_SERVER_SPEC_CONN | Connection | SQL_CONNECT_AS NULL |
| \$3: SQL_SERVER_SPEC_USER | User | SQL_CONNECT_USER NULL |

| | | |
|---------------------------|----------------------|---------------------------------|
| SQL_CONNECT_AS | Connection Specifier | AS \$1 |
| \$1: SQL_CONNECT_AS__NAME | Name | STR_CONST SQL_HOST_VAR_REF |

| | | |
|-----------------------------|----------------|---------------------------------|
| SQL_CONNECT_USER | User Specifier | USER \$1 |
| \$1: SQL_CONNECT_USER__NAME | Name | STR_CONST SQL_HOST_VAR_REF |

10.9.5 SELECT Statement

| | | |
|----------------------------------|---------------------|---|
| SQL_STMT_SELECT | Select Statement | SELECT{\$1? \$1}{\$2= \$2}{\$3= INTO \$3}{ |
| \$1: SEL_STMT_ALL_DISTINCT | All/Distinct clause | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SEL_STMT_ITEM_LIST | Item List | SQL_SEL_ITEM_LIST_STAR SQL_SEL_ITEM_LIST (SQL_SEL_ITEM_EXPR SQL_SEL_ITEM_TABLE_STAR) |
| \$3: SEL_STMT_INTOLIST | Into List | SQL_TARGET_LIST (SQL_TARGET) |
| \$4: SEL_STMT_FROM_LIST | From List | SQL_TABLE_REF_LIST (SQL_TABLE_REF_ITEM) |
| \$5: SEL_STMT_WHERE_CLAUSE | Where Clause | SQL_WHERE NULL |
| \$6: SEL_STMT_GROUP_BY_CLAUSE | Group By Clause | SQL_SEL_GROUP_BY NULL |
| \$7: SEL_STMT_HAVING_CLAUSE | Having Clause | SQL_SEL_HAVING NULL |

| | | |
|--------------|---------------|-----|
| SQL_QUAL_ALL | All Qualifier | ALL |
|--------------|---------------|-----|

| | | |
|-------------------|--------------------|----------|
| SQL_QUAL_DISTINCT | Distinct Qualifier | DISTINCT |
|-------------------|--------------------|----------|

| | | |
|------------------------|-----------------------|---|
| SQL_SEL_ITEM_LIST_STAR | Star Select Item List | * |
|------------------------|-----------------------|---|

| | | |
|-------------------|--------------------|-----|
| SQL_SEL_ITEM_LIST | SQL Statement List | @,& |
|-------------------|--------------------|-----|

| | | |
|---------------------------------|------------------------|----------|
| SQL_SEL_ITEM_EXPR | Expression Select Item | \$1 |
| \$1: SQL_SEL_ITEM_EXPR__EXPR | Expression | sql-expr |

| | | |
|--|---------------------|---------------|
| SQL_SEL_ITEM_TABLE_STAR | Table.* Select Item | \$1.* |
| \$1: SQL_SEL_ITEM_TABLE_STAR__TABLE | Table | SQL_TABLE_REF |

| | | |
|-----------------|-----------------|-----|
| SQL_TARGET_LIST | SQL Target List | @,& |
|-----------------|-----------------|-----|

| | | |
|---------------------------|-------------------|--------------------------|
| SQL_TARGET | SQL variable spec | \$1{\$2? INDICATOR \$2} |
| \$1: SQL_TARGET_HOST_VAR | Host Variable | SQL_HOST_VAR_REF |
| \$2: SQL_TARGET_INDICATOR | Indicator | SQL_VAR_INDICATOR NULL |

| | | |
|---------------------------------|-------------------------|------------------------------|
| SQL_HOST_VAR_REF | Host Variable Reference | \\:\$1 |
| \$1: SQL_HOST_VAR_REF__NAMES | Variable Refernce | SQL_VAR_NAME_LIST (NAME_USE) |

| | | |
|-------------------|-------------------------|----|
| SQL_VAR_NAME_LIST | Host Variable Name List | @. |
|-------------------|-------------------------|----|

| | | |
|-----------------------------|--------------------|------------------|
| SQL_VAR_INDICATOR | Indicator Variable | \$1 |
| \$1: SQL_VAR_INDICATOR__VAR | Variable | SQL_HOST_VAR_REF |

| | | |
|--------------------|--------------------------|----|
| SQL_TABLE_REF_LIST | SQL Table Reference List | @, |
|--------------------|--------------------------|----|

| | | |
|----------------------------------|--------------------------|-----------------------------------|
| SQL_TABLE_REF_ITEM | Table Reference Item | \$1{\$2? \$2} |
| \$1: SQL_TABLE_REF_ITEM_TABLE | Table Reference | SQL_TABLE_REF SQL_TABLE_EXPR |
| \$2: SQL_TABLE_REF_ITEM_VAR | Range Variable Definiton | SQL_RANGE_VAR_DEF NULL |

| | | |
|--------------------------|-----------------|----------------|
| SQL_TABLE_EXPR | Table Expr | (\$1) |
| \$1: SQL_TABLE_EXPR__ARG | Table Reference | sql-table-expr |

| | | |
|-----------------------------|--------------------------|--------------|
| SQL_RANGE_VAR_DEF | Range Variable Definiton | {\$1=AS \$1} |
| \$1: SQL_RANGE_VAR_DEF__VAR | Variable | CHAR_STRING |

| | | |
|----------------------|--------------|-----------|
| SQL_WHERE | Where Clause | WHERE \$1 |
| \$1: SQL_WHERE__COND | Condition | sql-cond |

| | | |
|-----------------------------------|------------------------|--------------------------------|
| SQL_SEL_GROUP_BY | Select Group By Clause | GROUP BY \$1 |
| \$1: SQL_SEL_GROUP_BY__COLUMNS | Columns | SQL_COL_REF_LIST (SQL_COL_REF) |

| | | |
|------------------|-----------------------|-----|
| SQL_COL_REF_LIST | Column Reference List | @,& |
|------------------|-----------------------|-----|

| | | |
|----------------------------|------------------|----------------------|
| SQL_COL_REF | Column Reference | {\$1?\$1.}{\$2=\$2} |
| \$1: SQL_COL_REF_QUALIFIER | Column | SQL_TABLE_REF NULL |
| \$2: SQL_COL_REF_COLUMN | Column | CHAR_STRING |

| | | |
|---------------------------|----------------------|------------|
| SQL_SEL_HAVING | Select Having Clause | HAVING \$1 |
| \$1: SQL_SEL_HAVING__COND | Condition | sql-cond |

10.9.6 Data Modification Statements

| | | |
|---------------------------|------------------|---|
| SQL_STMT_INSERT | Insert Statement | INSERT INTO \$1{\$2= \$2} |
| \$1: SQL_STMT_INSERT_INTO | Into Table | SQL_TABLE_REF |
| \$2: SQL_STMT_INSERT_WHAT | Insert What | SQL_INSERT_COLS SQL_DEFAULT_VALUES |

| | | |
|---------------------------|---------------------|----------------------|
| SQL_INSERT_COLS | Insert Table Expr | \$1{\$2= \$2} |
| \$1: SQL_INSERT_COLS_LIST | Columns List | SQL_OP_PARENS NULL |
| \$2: SQL_INSERT_COLS_EXPR | Inserted Table Expr | sql-table-expr |

| | | |
|--------------------|----------------|----------------|
| SQL_DEFAULT_VALUES | Default Values | DEFAULT VALUES |
|--------------------|----------------|----------------|

| | | |
|---------------------------|------------------|----------------------|
| SQL_STMT_DELETE | Delete Statement | DELETE \$1{\$2= \$2} |
| \$1: SQL_STMT_DELETE_FROM | From Table | SQL_FROM_TABLE |
| \$2: SQL_STMT_DELETE_COND | Where Clause | SQL_WHERE NULL |

| | | |
|---------------------------|------------------|---|
| SQL_STMT_UPDATE | Update Statement | UPDATE \$1{\$2= SET \$2}{\$3? \$3} |
| \$1: SQL_STMT_UPDATE_WHAT | Table | SQL_TABLE_REF |
| \$2: SQL_STMT_UPDATE_SET | Set Values | SQL_UPDATE_ASSIGN_LIST (SQL_UPDATE_ASSIGN) |
| \$3: SQL_STMT_UPDATE_COND | Where Clause | SQL_WHERE NULL |

| | | |
|------------------------|------------------------|-----|
| SQL_UPDATE_ASSIGN_LIST | Update Assingment List | @,& |
|------------------------|------------------------|-----|

| | | |
|----------------------------------|-------------------|-----------------|
| SQL_UPDATE_ASSIGN | Update Assignmnet | \$1{\$2= = \$2} |
| \$1: SQL_UPDATE_ASSIGN_COLUMN | Column | SQL_COL_REF |
| \$2: SQL_UPDATE_ASSIGN_VALUE | Value | sql-expr |

10.9.7 Cursor Statements

| | | |
|---|--------------------------|---|
| SQL_STMT_DECLARE_CURSOR | Declare Cursor Statement | DECLARE \$1{\$2? \$2{\$3? \$3} CURSOR{\$4= |
| \$1: SQL_STMT_DECLARE_CURSOR_CURSOR | Insensitive | SQL_DECL_INSENSITIVE NULL |
| \$2: SQL_STMT_DECLARE_CURSOR_INSENSITIVE | Scroll | SQL_DECL_SCROLL NULL |
| \$3: SQL_STMT_DECLARE_CURSOR_SCROLL | Cursor Defined | SQL_CURSOR_REF |
| \$4: SQL_STMT_DECLARE_CURSOR_TAIL | Declaration Tail | SQL_DECL_CURSOR_STATIC SQL_DECL_CURSOR_DYNAMIC |

| | | |
|--------------------------|------------------|-------------|
| SQL_CURSOR_REF | Cursor Reference | \$1 |
| \$1: SQL_CURSOR_REF_NAME | Cursor | CHAR_STRING |

| | | |
|----------------------|--------------------|-------------|
| SQL_DECL_INSENSITIVE | Insensitive Phrase | INSENSITIVE |
|----------------------|--------------------|-------------|

| | | |
|-----------------|---------------|--------|
| SQL_DECL_SCROLL | Scroll Phrase | SCROLL |
|-----------------|---------------|--------|

| | | |
|-----------------------------------|-----------------------|---|
| SQL_DECL_CURSOR_STATIC | Declare Cursor Static | { \$1= \$1 } { \$2? \$2 } { \$3? \$3 } |
| \$1: SQL_DECL_CUR_STATIC_EXPR | Table Expression | sql-table-expr |
| \$2: SQL_DECL_CUR_STATIC_ORDER | Order By | SQL_ORDER_BY NULL |
| \$3: SQL_DECL_CUR_STATIC_FOR | For | SQL_DECL_FOR NULL |

| | | |
|------------------------|-----------------|---|
| SQL_ORDER_BY | Order By Clause | ORDER BY \$1 |
| \$1: SQL_ORDER_BY_LIST | Cursor | SQL_ORDER_ITEM_LIST (SQL_ORDER_ITEM) |

| | | |
|---------------------|-----------------|-----|
| SQL_ORDER_ITEM_LIST | Order Item List | @,& |
|---------------------|-----------------|-----|

| | | |
|----------------------------------|------------|--|
| SQL_ORDER_ITEM | Order Item | \$1{\$2? \$2} |
| \$1: SQL_ORDER_ITEM_COL | Cursor | SQL_COL_REF INT_NUM_CONST |
| \$2: SQL_ORDER_ITEM_DIRECTION | Direction | SQL_ORDER_ASC SQL_ORDER_DESC NULL |

| | | |
|---------------|---------------------|-----|
| SQL_ORDER_ASC | Ascending Direction | ASC |
|---------------|---------------------|-----|

| | | |
|----------------|-----------------------|------|
| SQL_ORDER_DESC | Desscending Direction | DESC |
|----------------|-----------------------|------|

| | | |
|---------------------------|------------|--|
| SQL_DECL_FOR | For Phrase | FOR \$1 |
| \$1: SQL_DECL_FOR__ACCESS | Access | SQL_DECL_ACC_READONLY SQL_DECL_ACC_UPDATE |

| | | |
|-----------------------|------------------|-----------|
| SQL_DECL_ACC_READONLY | Read Only Access | READ ONLY |
|-----------------------|------------------|-----------|

| | | |
|-----------------------------------|----------------|-------------------------------|
| SQL_DECL_ACC_UPDATE | Update Access | UPDATE{\$1? \$1} |
| \$1: SQL_DECL_ACC_UPDATE__WHAT | Update Of What | SQL_DECL_ACC_UPDATE_OF NULL |

| | | |
|--------------------------------------|-------------|--------------------------------|
| SQL_DECL_ACC_UPDATE_OF | Update Of | OF \$1 |
| \$1: SQL_DECL_ACC_UPDATE_OF__COLS | Column List | SQL_COL_REF_LIST (SQL_COL_REF) |

| | | |
|---|------------------------|-----------------|
| SQL_DECL_CURSOR_DYNAMIC | Declare Cursor Dynamic | {\$1= \$1} |
| \$1: SQL_DECL_CURSOR_DYNAMIC__PREPARED | Prepared | SQL_PREPPED_REF |

| | | |
|-------------------------------------|-----------------------|-----------------------|
| SQL_STMT_OPEN_CURSOR | Open Cursor Statement | OPEN \$1{\$2? \$2} |
| \$1: SQL_STMT_OPEN_CURSOR_CURSOR | Cursor | SQL_CURSOR_REF |
| \$2: SQL_STMT_OPEN_CURSOR_USING | Using Args | SQL_USING_VARS NULL |

| | | |
|---------------------------------------|------------------------|----------------|
| SQL_STMT_CLOSE_CURSOR | Close Cursor Statement | CLOSE \$1 |
| \$1: SQL_STMT_CLOSE_CURSOR__CURSOR | Cursor | SQL_CURSOR_REF |

| | | |
|--------------------------------------|------------------------|--|
| SQL_STMT_FETCH_CURSOR | Fetch Cursor Statement | FETCH{\$1? \$1}{\$2= \$2}{\$3= \$3} |
| \$1: SQL_STMT_FETCH_CURSOR_ROWSEL | Row Selector | SQL_ROWSEL_NEXT SQL_ROWSEL_PRIOR SQL_ROWSEL_FIRST SQL_ROWSEL_LAST SQL_ROWSEL_ABSOLUTE SQL_ROWSEL_RELATIVE |
| \$2: SQL_STMT_FETCH_CURSOR_CURSOR | Cursor | SQL_CURSOR_REF |
| \$3: SQL_STMT_FETCH_CURSOR_INT0 | Into Places | SQL_INT0_VARS SQL_INT0_DESCR |

| | | |
|-----------------|-------------------|------|
| SQL_ROWSEL_NEXT | Next Row Selector | NEXT |
|-----------------|-------------------|------|

| | | |
|------------------|--------------------|-------|
| SQL_ROWSEL_PRIOR | Prior Row Selector | PRIOR |
|------------------|--------------------|-------|

| | | |
|------------------|--------------------|-------|
| SQL_ROWSEL_FIRST | First Row Selector | FIRST |
|------------------|--------------------|-------|

| | | |
|-----------------|-------------------|------|
| SQL_ROWSEL_LAST | Last Row Selector | LAST |
|-----------------|-------------------|------|

| | | |
|----------------------------------|-----------------------|---------------|
| SQL_ROWSEL_ABSOLUTE | Absolute Row Selector | ABSOLUTE \$1 |
| \$1: SQL_ROWSEL_ABSOLUTE_POSN | Position | INT_NUM_CONST |

| | | |
|----------------------------------|-----------------------|---------------|
| SQL_ROWSEL_RELATIVE | Relative Row Selector | RELATIVE \$1 |
| \$1: SQL_ROWSEL_RELATIVE_POSN | Position | INT_NUM_CONST |

| | | | |
|---------------------------------------|-------------------------|-------------------------------------|----|
| SQL_STMT_DELETE_CURSOR | Delete Cursor Statement | DELETE{\$1? \$1}{\$2= WHERE CURRENT | OF |
| \$1: SQL_STMT_DELETE_CURSOR_TABLE | From Table | SQL_FROM_TABLE NULL | |
| \$2: SQL_STMT_DELETE_CURSOR_CURSOR | At Cursor | SQL_CURSOR_REF | |

| | | |
|---------------------------------------|-------------------------|---|
| SQL_STMT_UPDATE_CURSOR | Update Cursor Statement | UPDATE{\$1? \$1}{\$2= SET \$2}{\$3= WHERE |
| \$1: SQL_STMT_UPDATE_CURSOR_TABLE | Table | SQL_TABLE_REF NULL |
| \$2: SQL_STMT_UPDATE_CURSOR_SET | Set What | SQL_UPDATE_ASSIGN_LIST (SQL_UPDATE_ASSIGN) |
| \$3: SQL_STMT_UPDATE_CURSOR_CURSOR | At Cursor | SQL_CURSOR_REF |

| | | |
|-------------------------|-------------------|---------------|
| SQL_FROM_TABLE | From Table Phrase | FROM \$1 |
| \$1: SQL_FROM_TABLE_REF | Table | SQL_TABLE_REF |

| | | |
|-------------------------|-----------------------|------------------------------|
| SQL_INTRO_VARS | Into Variables Phrase | INTO \$1 |
| \$1: SQL_INTRO_VARS_REF | Variables | SQL_TARGET_LIST (SQL_TARGET) |

| | | |
|--------------------------|------------------------|-------------------------|
| SQL_INTRO_DESCR | Into Descriptor Phrase | INTO SQL DESCRIPTOR \$1 |
| \$1: SQL_INTRO_DESCR_REF | Descriptor | SQL_DESCR_REF |

| | | |
|-------------------------|------------------------|------------------------------|
| SQL_USING_VARS | Using Variables Phrase | USING \$1 |
| \$1: SQL_USING_VARS_REF | Variables | SQL_TARGET_LIST (SQL_TARGET) |

| | | |
|--------------------------|-------------------------|--------------------------|
| SQL_USING_DESCR | Using Descriptor Phrase | USING SQL DESCRIPTOR \$1 |
| \$1: SQL_USING_DESCR_REF | Descriptor | SQL_DESCR_REF |

10.9.8 Dynamic SQL Statements

| | | |
|--|-------------------------------|-------------------------|
| SQL_STMT_ALLOCATE_DESCRIPTOR | Allocate Descriptor Statement | ALLOCATE DESCRIPTOR \$1 |
| \$1: SQL_STMT_ALLOCATE_DESCRIPTOR_REF | Descriptor | SQL_DESCR_REF |

| | | |
|--|---------------------------|-----------------------------------|
| SQL_STMT_ALLOCATE_CURSOR | Allocate Cursor Statement | ALLOCATE \$1 CURSOR{\$2= FOR \$2} |
| \$1: SQL_STMT_ALLOC_CURSOR_CURSOR | Cursor | SQL_CURSOR_REF |
| \$2: SQL_STMT_ALLOC_CURSOR_PREPARED | Prepared | SQL_PREPPED_REF |

| | | |
|--|---------------------------------|---------------------------|
| SQL_STMT DEALLOCATE_DESCRIPTOR | Deallocate Descriptor Statement | DEALLOCATE DESCRIPTOR \$1 |
| \$1: SQL_STMT DEALLOCATE_DESCRIPTOR_REF | Descriptor | SQL_DESCR_REF |

| | | |
|---|------------------------------|------------------------|
| SQL_STMT_DEALLOCATE_PREPARE | Deallocate Prepare Statement | DEALLOCATE PREPARE \$1 |
| \$1: SQL_STMT_DEALLOCATE_PREPARE_REF | Prepared REF | SQL_PREPPED_REF |

| | | |
|--|--------------------------|---|
| SQL_STMT_DESCRIBE_INPUT | Describe Input Statement | DESCRIBE INPUT \$1{\$2= USING SQL_DESCR |
| \$1: SQL_STMT_DESCRIBE_INPUT__PREPPED | Prepared | SQL_PREPPED_REF |
| \$2: SQL_STMT_DESCRIBE_INPUT__DESCR | Descriptor | SQL_DESCR_REF |

| | | |
|---------------------------------------|---------------------------|--|
| SQL_STMT_DESCRIBE_OUTPUT | Describe Output Statement | DESCRIBE OUTPUT \$1{\$2= USING SQL_DESCR |
| \$1: SQL_STMT_DESCR_OUTPUT_PREPPED | Prepared | SQL_PREPPED_REF |
| \$2: SQL_STMT_DESCR_OUTPUT_DESCR | Descriptor | SQL_DESCR_REF |

| | | |
|---|-----------------------------|-----------------------|
| SQL_STMT_EXECUTE_IMMEDIATE | Execute Immediate Statement | EXECUTE IMMEDIATE \$1 |
| \$1: SQL_STMT_EXECUTE_IMMEDIATE_TEXT | Text To Execute TEXT | SQL_HOST_VAR_REF |

| | | |
|------------------------------------|----------------------------|---------------------------------|
| SQL_STMT_EXECUTE_PREPARED | Execute Prepared Statement | EXECUTE \$1{\$2? \$2}{\$3? \$3} |
| \$1: SQL_STMT_EXEC_PREP_PREPPED | Prepared | SQL_PREPPED_REF |
| \$2: SQL_STMT_EXEC_PREP_INT0 | Into Places | SQL_PREPPED_REF |
| \$3: SQL_STMT_EXEC_PREP_USING | Using Args | SQL_PREPPED_REF |

| | | |
|----------------------------------|-------------------|----------------------------|
| SQL_STMT_PREPARE | Prepare Statement | PREPARE \$1{\$2= FROM \$2} |
| \$1: SQL_STMT_PREPARE_PREPPED | Prepared | SQL_PREPPED_REF |
| \$2: SQL_STMT_PREPARE_FROM | From Variable | SQL_HOST_VAR_REF |

| | | |
|--------------------------|----------------------|----------------------------|
| SQL_DESCR_REF | Descriptor Reference | \$1 |
| \$1: SQL_DESCR_REF__NAME | Descriptor Name | STR_CONST CHAR_STRING |

| | | |
|----------------------------|--------------------|-----------------------------------|
| SQL_PREPPED_REF | Prepared Reference | \$1 |
| \$1: SQL_PREPPED_REF__NAME | Prepared Name | SQL_HOST_VAR_REF CHAR_STRING |

| | | |
|---------------------------------|--------------------------|-----------------------------------|
| SQL_STMT_GET_DESCRIPTOR | Get Descriptor Statement | GET_DESCRIPTOR \$1{\$2= \$2} |
| \$1: SQL_STMT_GET_DESCR_REF | Descriptor | SQL_DESCR_REF |
| \$2: SQL_STMT_GET_DESCR_ARGS | Get Args | SQL_GET_COUNT SQL_GET_VALUES |

| | | |
|----------------------------|-----------------|-------------|
| SQL_GET_COUNT | Get Count | \$1 = COUNT |
| \$1: SQL_GET_COUNT__TARGET | Target Variable | SQL_TARGET |

| | | |
|----------------------------|--------------|---|
| SQL_GET_VALUES | Get Values | \$1{\$2= \$2} |
| \$1: SQL_GET_VALUES_NUMBER | Number | INT_NUM_CONST CHAR_STRING |
| \$2: SQL_GET_VALUES_VALUES | Values Array | SQL_GET_ASSIGN_LIST (SQL_GET_ASSIGN) |

| | | |
|---------------------|-----------------|-----|
| SQL_GET_ASSIGN_LIST | Get Assign List | @,& |
|---------------------|-----------------|-----|

| | | |
|----------------------------|-----------------|-----------------|
| SQL_GET_ASSIGN | Get Assign Item | \$1 = {\$2=\$2} |
| \$1: SQL_GET_ASSIGN_TARGET | Target | SQL_TARGET |
| \$2: SQL_GET_ASSIGN_VALUE | Value | CHAR_STRING |

| | | |
|---------------------------------|--------------------------|-----------------------------------|
| SQL_STMT_SET_DESCRIPTOR | Set Descriptor Statement | SET_DESCRIPTOR \$1{\$2= \$2} |
| \$1: SQL_STMT_SET_DESCR_REF | Descriptor | SQL_DESCR_REF |
| \$2: SQL_STMT_SET_DESCR_ARGS | Set Args | SQL_SET_COUNT SQL_SET_VALUES |

| | | |
|----------------------------|-----------|-------------|
| SQL_SET_COUNT | Set Count | \$1 = COUNT |
| \$1: SQL_SET_COUNT__TARGET | Target | SQL_TARGET |

| | | |
|----------------------------|--------------|---|
| SQL_SET_VALUES | Set Values | \$1{\$2= \$2} |
| \$1: SQL_SET_VALUES_NUMBER | Number | INT_NUM_CONST CHAR_STRING |
| \$2: SQL_SET_VALUES_VALUES | Values Array | SQL_SET_ASSIGN_LIST (SQL_SET_ASSIGN) |

| | | |
|---------------------|-----------------|-----|
| SQL_SET_ASSIGN_LIST | Set Assign List | @,& |
|---------------------|-----------------|-----|

| | | |
|----------------------------|-----------------|-----------------|
| SQL_SET_ASSIGN | Set Assign Item | \$1 = {\$2=\$2} |
| \$1: SQL_SET_ASSIGN_VALUE | Value | CHAR_STRING |
| \$2: SQL_SET_ASSIGN_TARGET | Target | SQL_TARGET |

10.9.9 Expression

| | | |
|--|-----------------------|---|
| <ul style="list-style-type: none"> sql-expr | SQL Scalar Expression | INT_NUM_CONST REAL_NUM_CONST STR_CONST SQL_VALUE_DEFAULT SQL_VALUE_NULL SQL_OP_ADD (sql-expr) SQL_OP_SUB (sql-expr) SQL_OP_MUL (sql-expr) SQL_OP_DIV (sql-expr) SQL_OP_CONCAT (sql-expr) SQL_OP_UN_PLUS SQL_OP_UN_MINUS SQL_OP_PARENS SQL_SF_BIT_LENGTH SQL_SF_OCTET_LENGTH SQL_SF_CHAR_LENGTH SQL_SF_CURRENT_USER SQL_SF_SESSION_USER SQL_SF_SYSTEM_USER SQL_SF_LOWER SQL_SF_UPPER SQL_SF_POSITION SQL_SF_SUBSTRING SQL_SF_TRIM SQL_SF_CASE SQL_AF_COUNT_STAR SQL_AF_COUNT SQL_AF_AVG SQL_AF_MAX SQL_AF_MIN SQL_AF_SUM |
|--|-----------------------|---|

| | | |
|-------------------|---------------|---------|
| SQL_VALUE_DEFAULT | Default Value | DEFAULT |
|-------------------|---------------|---------|

| | | |
|----------------|------------|------|
| SQL_VALUE_NULL | Null Value | NULL |
|----------------|------------|------|

| | | |
|----------------------------------|---------------------------|--------------------|
| SQL_OP_ADD | SQL Add | @ +& |
| SQL_OP_SUB | SQL Subtract | @ -& |
| SQL_OP_MUL | SQL Multiply | @ *& |
| SQL_OP_DIV | SQL Divide | @ /& |
| SQL_OP_CONCAT | SQL Concatenate | @ & |
| SQL_OP_UN_MINUS | SQL Unary Minus | - \$1 |
| \$1: SQL_OP_UN_MINUS__ARG | Arg | sql-expr |
| SQL_OP_UN_PLUS | SQL Unary Plus | + \$1 |
| \$1: SQL_OP_UN_PLUS__ARG | Arg | sql-expr |
| SQL_OP_PARENS | SQL Parentheses | (\$1) |
| \$1: SQL_OP_PARENS__ARG | Arg | sql-expr |
| SQL_SF_BIT_LENGTH | SQL Bit Length Function | BIT_LENGTH(\$1) |
| \$1: SQL_SF_BIT_LENGTH__ARG | Arg | sql-expr |
| SQL_SF_OCTET_LENGTH | SQL Octet Length Function | OCTET_LENGTH(\$1) |
| \$1: SQL_SF_OCTET_LENGTH__ARG | Arg | sql-expr |
| SQL_SF_CHAR_LENGTH | SQL Char Length Function | CHAR_LENGTH(\$1) |
| \$1: SQL_SF_CHAR_LENGTH__ARG | Arg | sql-expr |
| SQL_SF_CURRENT_USER | SQL Current User Function | CURRENT_USER |
| SQL_SF_SESSION_USER | SQL Session User Function | SESSION_USER |
| SQL_SF_SYSTEM_USER | SQL System User Function | SYSTEM_USER |

| | | |
|-----------------------|--------------------|------------|
| SQL_SF_LOWER | SQL Lower Function | LOWER(\$1) |
| \$1: SQL_SF_LOWER_ARG | Arg | sql-expr |

| | | |
|-----------------------|--------------------|------------|
| SQL_SF_UPPER | SQL Upper Function | UPPER(\$1) |
| \$1: SQL_SF_UPPER_ARG | Arg | sql-expr |

| | | |
|---------------------------|-----------------------|----------------------|
| SQL_SF_POSITION | SQL Position Function | POSITION(\$1 IN \$2) |
| \$1: SQL_SF_POSITION_ARG1 | Arg 1 | sql-expr |
| \$2: SQL_SF_POSITION_ARG2 | Arg 2 | sql-expr |

| | | |
|----------------------------|------------------------|---------------------------------|
| SQL_SF_SUBSTRING | SQL Substring Function | SUBSTRING(\$1 FROM \$2 FOR \$3) |
| \$1: SQL_SF_SUBSTRING_ARG1 | Arg 1 | sql-expr |
| \$2: SQL_SF_SUBSTRING_ARG2 | Arg 2 | sql-expr |
| \$3: SQL_SF_SUBSTRING_ARG3 | Arg 3 | sql-expr |

| | | |
|-----------------------|-------------------|--|
| SQL_SF_TRIM | SQL Trim Function | TRIM(\$1 \$2 FROM \$3) |
| \$1: SQL_SF_TRIM_ARG1 | Arg 1 | SQL_TRIM_LEADING SQL_TRIM_TRAILING SQL_TRIM_BOTH |
| \$2: SQL_SF_TRIM_ARG2 | Arg 2 | sql-expr |
| \$3: SQL_SF_TRIM_ARG3 | Arg 3 | sql-expr |

| | | |
|------------------|--------------|---------|
| SQL_TRIM_LEADING | Leading Trim | LEADING |
|------------------|--------------|---------|

| | | |
|-------------------|---------------|----------|
| SQL_TRIM_TRAILING | Trailing Trim | TRAILING |
|-------------------|---------------|----------|

| | | |
|---------------|-----------|------|
| SQL_TRIM_BOTH | Both Trim | BOTH |
|---------------|-----------|------|

| | | |
|------------------------|-------------------|---------------------------------------|
| SQL_SF_CASE | SQL Case Function | CASE \$1 ELSE \$2 END |
| \$1: SQL_SF_CASE_WHENS | When Clauses | SQL_WHEN_EXPR_LIST (SQL_WHEN_EXPR) |
| \$2: SQL_SF_CASE_ELSE | Else Clause | sql-expr |

| | | |
|--------------------|--------------------------|----|
| SQL_WHEN_EXPR_LIST | SQL When Expression List | @& |
|--------------------|--------------------------|----|

| | | |
|-------------------------|---------------------|------------------------|
| SQL_WHEN_EXPR | SQL When Expression | WHEN \$1{\$2=THEN \$2} |
| \$1: SQL_WHEN_EXPR_COND | Condition | sql-expr |
| \$2: SQL_WHEN_EXPR_RET | Return Value | sql-expr |

| | | |
|-------------------|-------------------|----------|
| SQL_AF_COUNT_STAR | Count(*) Function | COUNT(*) |
|-------------------|-------------------|----------|

| | | |
|----------------------------|----------------|--|
| SQL_AF_COUNT | Count Function | COUNT({\$1?\$1 }{\$2=\$2}) |
| \$1: SQL_AF_COUNT_ALL_DIST | All/Distinct | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SQL_AF_COUNT_ARG | Arg | sql-expr |

| | | |
|--------------------------|------------------|--|
| SQL_AF_AVG | Average Function | AVG({\$1?\$1 }{\$2=\$2}) |
| \$1: SQL_AF_AVG_ALL_DIST | All/Distinct | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SQL_AF_AVG_ARG | Arg | sql-expr |

| | | |
|--------------------------|--------------|--|
| SQL_AF_MAX | MAX Function | MAX({\$1?\$1 }{\$2=\$2}) |
| \$1: SQL_AF_MAX_ALL_DIST | All/Distinct | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SQL_AF_MAX_ARG | Arg | sql-expr |

| | | |
|--------------------------|--------------|--|
| SQL_AF_MIN | MIN Function | MIN({\$1?\$1 }{\$2=\$2}) |
| \$1: SQL_AF_MIN_ALL_DIST | All/Distinct | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SQL_AF_MIN_ARG | Arg | sql-expr |

| | | |
|--------------------------|--------------|--|
| SQL_AF_SUM | SUM Function | SUM({\$1?\$1 }{\$2=\$2}) |
| \$1: SQL_AF_SUM_ALL_DIST | All/Distinct | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SQL_AF_SUM_ARG | Arg | sql-expr |

10.9.10 Condition

| | | |
|------------|---------------|---|
| • sql-cond | SQL Condition | SQL_OP_OR (sql-cond) SQL_OP_AND (sql-cond) SQL_OP_NOT SQL_OP_EQ SQL_OP_NE SQL_OP_GT SQL_OP_GE SQL_OP_LT SQL_OP_LE SQL_OP_BETWEEN SQL_OP_NOT_BETWEEN SQL_OP_IN SQL_OP_NOT_IN SQL_OP_LIKE SQL_OP_NOT_LIKE SQL_OP_IS_NULL SQL_OP_IS_NOT_NULL SQL_OP_EXISTS SQL_OP_UNIQUE |
|------------|---------------|---|

| | | |
|-----------|--------|-------|
| SQL_OP_OR | SQL OR | @ OR& |
|-----------|--------|-------|

| | | |
|------------|---------|--------|
| SQL_OP_AND | SQL AND | @ AND& |
|------------|---------|--------|

| | | |
|---------------------|-------|----------|
| SQL_OP_NOT | Equal | NOT \$1 |
| \$1: SQL_OP_NOT_ARG | Arg | sql-cond |

| | | |
|---------------------|-------|---------------|
| SQL_OP_EQ | Equal | \$1{\$2= \$2} |
| \$1: SQL_OP_EQ_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_EQ_ARG2 | Arg2 | sql-expr |

| | | |
|---------------------|-----------|-----------------|
| SQL_OP_NE | Not Equal | \$1{\$2= < \$2} |
| \$1: SQL_OP_NE_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_NE_ARG2 | Arg2 | sql-expr |

| | | |
|---------------------|--------------|-----------------|
| SQL_OP_GT | Greater Than | \$1{\$2= > \$2} |
| \$1: SQL_OP_GT_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_GT_ARG2 | Arg2 | sql-expr |

| | | |
|---------------------|----------------------|------------------|
| SQL_OP_GE | Greter Than Or Equal | \$1{\$2= >= \$2} |
| \$1: SQL_OP_GE_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_GE_ARG2 | Arg2 | sql-expr |

| | | |
|---------------------|-----------|-----------------|
| SQL_OP_LT | Less Than | \$1{\$2= < \$2} |
| \$1: SQL_OP_LT_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_LT_ARG2 | Arg2 | sql-expr |

| | | |
|---------------------|--------------------|------------------|
| SQL_OP_LE | Less Than Or Equal | \$1{\$2= <= \$2} |
| \$1: SQL_OP_LE_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_LE_ARG2 | Arg2 | sql-expr |

| | | |
|--------------------------|---------|-------------------------------------|
| SQL_OP_BETWEEN | Between | \$1{\$2= BETWEEN \$2}{\$3= AND \$3} |
| \$1: SQL_OP_BETWEEN_WHAT | What | sql-expr |
| \$2: SQL_OP_BETWEEN_FROM | From | sql-expr |
| \$3: SQL_OP_BETWEEN_TO | To | sql-expr |

| | | |
|---------------------------------|---------|---|
| SQL_OP_NOT_BETWEEN | Between | \$1{\$2= NOT BETWEEN \$2}{\$3= AND \$3} |
| \$1: SQL_OP_NOT_BETWEEN_WHAT | What | sql-expr |
| \$2: SQL_OP_NOT_BETWEEN_FROM | From | sql-expr |
| \$3: SQL_OP_NOT_BETWEEN_TO | To | sql-expr |

| | | |
|---------------------|---------|------------------|
| SQL_OP_IN | Between | \$1{\$2= IN \$2} |
| \$1: SQL_OP_IN_FROM | From | sql-expr |
| \$2: SQL_OP_IN_TO | To | sql-expr |

| | | |
|-------------------------|---------|----------------------|
| SQL_OP_NOT_IN | Between | \$1{\$2= NOT IN \$2} |
| \$1: SQL_OP_NOT_IN_FROM | From | sql-expr |
| \$2: SQL_OP_NOT_IN_TO | To | sql-expr |

| | | |
|-----------------------|---------|--------------------|
| SQL_OP_LIKE | Between | \$1{\$2= LIKE \$2} |
| \$1: SQL_OP_LIKE_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_LIKE_ARG2 | Arg2 | sql-expr |

| | | |
|---------------------------|---------|------------------------|
| SQL_OP_NOT_LIKE | Between | \$1{\$2= NOT LIKE \$2} |
| \$1: SQL_OP_NOT_LIKE_ARG1 | Arg1 | sql-expr |
| \$2: SQL_OP_NOT_LIKE_ARG2 | Arg2 | sql-expr |

| | | |
|-------------------------|---------|-------------|
| SQL_OP_IS_NULL | Is Null | \$1 IS NULL |
| \$1: SQL_OP_IS_NULL_ARG | Arg | sql-expr |

| | | |
|--------------------------------|---------|-------------|
| SQL_OP_IS_NOT_NULL | Is Null | \$1 IS NULL |
| \$1: SQL_OP_IS_NOT_NULL_ARG | Arg | sql-expr |

| | | |
|------------------------|--------|-------------|
| SQL_OP_EXISTS | Exists | EXISTS(\$1) |
| \$1: SQL_OP_EXISTS_ARG | Arg | sql-expr |

| | | |
|------------------------|--------|-------------|
| SQL_OP_UNIQUE | Unique | UNIQUE(\$1) |
| \$1: SQL_OP_UNIQUE_ARG | Arg | sql-expr |

10.9.11 Table Expression (Subquery)

| | | |
|------------------|----------------------|---|
| • sql-table-expr | SQL Table Expression | SQL_OP_JOIN SQL_OP_CROSS_JOIN SQL_OP_UNION SQL_OP_EXCEPT SQL_OP_INTERSECT SQL_OP_PARENS SQL_OP_VALUES SQL_EXPR_SELECT SQL_OP_TABLE SQL_TABLE_REF |
|------------------|----------------------|---|

| | | |
|---------------------------|----------------|--|
| SQL_OP_JOIN | Join Operation | \$1{\$2? \$2}{\$3= \$3} JOIN{\$4= \$4}{\$5? \$5} |
| \$1: SQL_OP_JOIN_ARG1 | Arg1 | sql-table-expr |
| \$2: SQL_OP_JOIN_NATURAL | Natural | SQL_JOIN_NATURAL NULL |
| \$3: SQL_OP_JOIN_TYPE | Type | SQL_JOIN_INNER SQL_JOIN_LEFT SQL_JOIN_LEFT_OUTER SQL_JOIN_RIGHT SQL_JOIN_RIGHT_OUTER SQL_JOIN_UNION |
| \$4: SQL_OP_JOIN_ARG2 | Arg2 | sql-table-expr |
| \$5: SQL_OP_JOIN_ON_USING | On/Using | SQL_JOIN_ON SQL_JOIN_USING NULL |

| | | |
|------------------|--------------|---------|
| SQL_JOIN_NATURAL | Natural Join | NATURAL |
|------------------|--------------|---------|

| | | |
|----------------|------------|-------|
| SQL_JOIN_INNER | Inner Join | INNER |
|----------------|------------|-------|

| | | |
|---------------|-----------|------|
| SQL_JOIN_LEFT | Left Join | LEFT |
|---------------|-----------|------|

| | | |
|---------------------|-----------------|------------|
| SQL_JOIN_LEFT_OUTER | Left Outer Join | LEFT OUTER |
|---------------------|-----------------|------------|

| | | |
|----------------|------------|-------|
| SQL_JOIN_RIGHT | Right Join | RIGHT |
|----------------|------------|-------|

| | | |
|----------------------|------------------|-------------|
| SQL_JOIN_RIGHT_OUTER | Right Outer Join | RIGHT OUTER |
|----------------------|------------------|-------------|

| | | |
|----------------|------------|-------|
| SQL_JOIN_UNION | Union Join | UNION |
|----------------|------------|-------|

| | | |
|------------------------|----------------|----------|
| SQL_JOIN_ON | Join On Clause | ON \$1 |
| \$1: SQL_JOIN_ON__COND | Condition | sql-cond |

| | | |
|---------------------------|----------------|--------------------------------|
| SQL_JOIN_USING | Join On Clause | USING (\$1) |
| \$1: SQL_JOIN_USING__COLS | Columns | SQL_COL_REF_LIST (SQL_COL_REF) |

| | | |
|-----------------------------|----------------------|--------------------------|
| SQL_OP_CROSS_JOIN | Cross Join Operation | \$1 CROSS JOIN{\$2= \$2} |
| \$1: SQL_OP_CROSS_JOIN_ARG1 | Arg1 | sql-table-expr |
| \$2: SQL_OP_CROSS_JOIN_ARG2 | Arg2 | sql-table-expr |

| | | |
|------------------------|-----------------|---|
| SQL_OP_UNION | Union Operation | \$1 UNION{\$2? \$2}{\$3? \$3}{\$4= \$4} |
| \$1: SQL_OP_UNION_ARG1 | Arg1 | sql-table-expr |
| \$2: SQL_OP_UNION_ALL | All | SQL_ALL NULL |
| \$3: SQL_OP_UNION_CORR | Corresponding | SQL_CORRESPONDING NULL |
| \$4: SQL_OP_UNION_ARG2 | Arg2 | sql-table-expr |

| | | |
|-------------------------|------------------|--|
| SQL_OP_EXCEPT | Except Operation | \$1 EXCEPT{\$2? \$2}{\$3? \$3}{\$4= \$4} |
| \$1: SQL_OP_EXCEPT_ARG1 | Arg1 | sql-table-expr |
| \$2: SQL_OP_EXCEPT_ALL | All | SQL_ALL NULL |
| \$3: SQL_OP_EXCEPT_CORR | Corresponding | SQL_CORRESPONDING NULL |
| \$4: SQL_OP_EXCEPT_ARG2 | Arg2 | sql-table-expr |

| | | | |
|----------------------------|---------------------|---|------|
| SQL_OP_INTERSECT | Intersect Operation | \$1 INTERSECT{\$2? \$2}{\$3? \$3}{\$4= \$4} | \$4} |
| \$1: SQL_OP_INTERSECT_ARG1 | Arg1 | sql-table-expr | |
| \$2: SQL_OP_INTERSECT_ALL | All | SQL_ALL NULL | |
| \$3: SQL_OP_INTERSECT_CORR | Corresponding | SQL_CORRESPONDING NULL | |
| \$4: SQL_OP_INTERSECT_ARG2 | Arg2 | sql-table-expr | |

| | | |
|---------|------------|-----|
| SQL_ALL | All Phrase | ALL |
|---------|------------|-----|

| | | |
|----------------------------|----------------------|-------------------------|
| SQL_CORRESPONDING | Corresponding Clause | CORRESPONDING{\$1? \$1} |
| \$1: SQL_CORRESPONDING__BY | By | SQL_CORRESP_BY NULL |

| | | |
|---------------------------|-------------------------|--------------------------------|
| SQL_CORRESP_BY | Corresponding By Clause | BY{\$1=\$1} |
| \$1: SQL_CORRESP_BY__COLS | Columns | SQL_COL_REF_LIST (SQL_COL_REF) |

| | | |
|-------------------------|------------------|--|
| SQL_OP_VALUES | Values Operation | VALUES \$1 |
| \$1: SQL_OP_VALUES__ARG | Values List | SQL_ROW_CONSTRUCTOR_LIST (SQL_ROW_CONSTRUCTOR) |

| | | |
|--------------------------|----------------------|-----|
| SQL_ROW_CONSTRUCTOR_LIST | Row Constructor List | @,& |
|--------------------------|----------------------|-----|

| | | |
|----------------------------------|-------------------------------|---|
| SQL_ROW_CONSTRUCTOR | Parenthesized Row Constructor | (\$1) |
| \$1: SQL_ROW_CONSTRUCTOR_ARGS | Row Elements | SQL_EXPR_LIST (sql-expr sql-table-expr) |

| | | |
|---------------|-----------------|-----|
| SQL_EXPR_LIST | Expression List | @,& |
|---------------|-----------------|-----|

| | | |
|----------------------------------|---------------------|---|
| SQL_EXPR_SELECT | Select Statement | SELECT{\$1? \$1}{\$2= \$2}{\$3= FROM \$3}{ |
| \$1: SEL_EXPR_ALL_DISTINCT | All/Distinct clause | SQL_QUAL_ALL SQL_QUAL_DISTINCT NULL |
| \$2: SEL_EXPR_ITEM_LIST | Item List | SQL_SEL_ITEM_LIST_STAR SQL_SEL_ITEM_LIST (SQL_SEL_ITEM_EXPR SQL_SEL_ITEM_TABLE_STAR) |
| \$3: SEL_EXPR_FROM_LIST | From List | SQL_TABLE_REF_LIST (SQL_TABLE_REF_ITEM) |
| \$4: SEL_EXPR_WHERE_CLAUSE | Where Clause | SQL_WHERE NULL |
| \$5: SEL_EXPR_GROUP_BY_CLAUSE | Group By Clause | SQL_SEL_GROUP_BY NULL |
| \$6: SEL_EXPR_HAVING_CLAUSE | Having Clause | SQL_SEL_HAVING NULL |

| | | |
|-----------------------|-----------------|---------------|
| SQL_OP_TABLE | Table Operation | TABLE \$1 |
| \$1: SQL_OP_TABLE_ARG | Table Ref | SQL_TABLE_REF |

| | | |
|---------------------------|---------------------|---------------------|
| SQL_TABLE_REF | SQL Table Reference | {\$1?\$1.}{\$2=\$2} |
| \$1: SQL_TABLE_REF_SCHEMA | Schema | CHAR_STRING NULL |
| \$2: SQL_TABLE_REF_TABLE | Table | CHAR_STRING |

10.10 CICS Extension

10.10.1 Top Level Statements

| | | |
|--------------------------|---------------------|------------------------------|
| STMT_EXEC_CICS | Exec Cics statement | EXEC CICS\n\t\$1\b\nEND-EXEC |
| \$1: STMT_EXEC_CICS_STMT | Statement | NULL |

| | | |
|---------------------|------------------------|--|
| CS_STMT_ABEND | Abend Statement (CICS) | ABEND\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
| \$1: ABEND_NOT_USED | not-used | NULL |
| \$2: ABEND_ABCODE | Abcode | CS_CL_ABCODE NULL |
| \$3: ABEND_CANCEL | Cancel | CS_CL_CANCEL NULL |
| \$4: ABEND_NODUMP | Nodump | CS_CL_NODUMP NULL |
| \$5: ABEND_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$6: ABEND_RESP | Resp | CS_CL_RESP NULL |
| \$7: ABEND_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------|--------------------------|--|
| CS_STMT_ADDRESS | Address Statement (CICS) | ADDRESS\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
| \$1: ADDR_NOT_USED | not-used | NULL |
| \$2: ADDR_ACEE | Acee | CS_CL_ACEE NULL |
| \$3: ADDR_COMMAREA | Commarea | CS_CL_COMMAREA NULL |
| \$4: ADDR_CWA | Cwa | CS_CL_CWA NULL |
| \$5: ADDR_EIB | Eib | CS_CL_EIB NULL |
| \$6: ADDR_TCTUA | Tctua | CS_CL_TCTUA NULL |
| \$7: ADDR_TWA | Twa | CS_CL_TWA NULL |
| \$8: ADDR_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$9: ADDR_RESP | Resp | CS_CL_RESP NULL |
| \$10: ADDR_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-----------------------|------------------------------|---|
| CS_STMT_ADDRESS_SET | Address Set Statement (CICS) | ADDRESS{\$1?\n\$1}{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: ADDRSET_SET | Set | CS_CL_SET |
| \$2: ADDRSET_USING | Using | CS_CL_USING |
| \$3: ADDRSET_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: ADDRSET_RESP | Resp | CS_CL_RESP NULL |
| \$5: ADDRSET_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-------------------|--------------------------|---|
| CS_STMT_ASKTIME | Asktime Statement (CICS) | ASKTIME\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: ASK_NOT_USED | not-used | NULL |
| \$2: ASK_ABSTIME | Abstime | CS_CL_ABSTIME NULL |
| \$3: ASK_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: ASK_RESP | Resp | CS_CL_RESP NULL |
| \$5: ASK_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------------|-------------------------|---|
| CS_STMT_ASSIGN | Assign Statement (CICS) | ASSIGN\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{ |
| \$1: ASS_NOT_USED | not-used | NULL |
| \$2: ASS_ASSIGNMENT_LIST | Assignment List | assignment-list |
| \$3: ASS_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: ASS_RESP | Resp | CS_CL_RESP NULL |
| \$5: ASS_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------|-----------------------------|---|
| CS_STMT_BIF_DEEDIT | Bif Deedit Statement (CICS) | BIF DEEDIT\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{ |
| \$1: BIF_NOT_USED | not-used | NULL |
| \$2: BIF_FIELD | Field | CS_CL_FIELD |
| \$3: BIF_LENGTH | Length | CS_CL_LENGTH NULL |
| \$4: BIF_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$5: BIF_RESP | Resp | CS_CL_RESP NULL |
| \$6: BIF_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------------|------------------------|--|
| CS_STMT_DELAY | Delay Statement (CICS) | DELAY\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{ |
| \$1: DELAY_NOT_USED | not-used | NULL |
| \$2: DELAY_INTERVAL_TIME | Interval/Time | CS_CL_INTERVAL CS_CL_TIME NULL |
| \$3: DELAY_MODE | Mode | CS_CL_AFTER CS_CL_AT CS_CL_FOR CS_CL_UNTIL NULL |
| \$4: DELAY_HOURS | Hours | CS_CL_HOURS NULL |
| \$5: DELAY_MINUTES | Minutes | CS_CL_MINUTES NULL |
| \$6: DELAY_SECONDS | Seconds | CS_CL_SECONDS NULL |
| \$7: DELAY_REQID | Reqid | CS_CL_REQID NULL |
| \$8: DELAY_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$9: DELAY_RESP | Resp | CS_CL_RESP NULL |
| \$10: DELAY_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-----------------------|-------------------------|---|
| CS_STMT_DELETE | Delete Statement (CICS) | DELETE\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{ |
| \$1: DEL_NOT_USED | not-used | NULL |
| \$2: DEL_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: DEL_TOKEN | Token | CS_CL_TOKEN NULL |
| \$4: DEL_RIDFLD | Ridfld | CS_CL_RIDFLD NULL |
| \$5: DEL_KEYLENGTH | Keylength | CS_CL_KEYLENGTH NULL |
| \$6: DEL_GENERIC | Generic | CS_CL_GENERIC NULL |
| \$7: DEL_NUMREC | Numrec | CS_CL_NUMREC NULL |
| \$8: DEL_SYSID | Sysid | CS_CL_SYSID NULL |
| \$9: DEL_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$10: DEL_RBA_RRN | Rba/Rrn | CS_CL_RBA CS_CL_RRN NULL |
| \$11: DEL_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$12: DEL_RESP | Resp | CS_CL_RESP NULL |
| \$13: DEL_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------|-----------------------------|--|
| CS_STMT_DELETEQ_TD | Deleteq Td Statement (CICS) | DELETEQ TD\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: DELTD_NOT_USED | not-used | NULL |
| \$2: DELTD_QUEUE | Queue | CS_CL_QUEUE |
| \$3: DELTD_SYSID | Sysid | CS_CL_SYSID NULL |
| \$4: DELTD_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$5: DELTD_RESP | Resp | CS_CL_RESP NULL |
| \$6: DELTD_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------|-----------------------------|--|
| CS_STMT_DELETEQ_TS | Deleteq Ts Statement (CICS) | DELETEQ TS\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: DELTS_NOT_USED | not-used | NULL |
| \$2: DELTS_QUEUE | Queue | CS_CL_QUEUE CS_CL_QNAME |
| \$3: DELTS_SYSID | Sysid | CS_CL_SYSID NULL |
| \$4: DELTS_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$5: DELTS_RESP | Resp | CS_CL_RESP NULL |
| \$6: DELTS_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------|-----------------------|---|
| CS_STMT_DUMP | Dump Statement (CICS) | DUMP\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
| \$1: DUMP_NOT_USED | not-used | NULL |
| \$2: DUMP_DUMPCODE | Dumpcode | CS_CL_DUMPCODE |
| \$3: DUMP_FROM | From | CS_CL_FROM NULL |
| \$4: DUMP_LENGTH | Length | CS_CL_LENGTH CS_CL_FLENGTH NULL |
| \$5: DUMP_COMPLETE | Complete | CS_CL_COMPLETE NULL |
| \$6: DUMP_TASK | Task | CS_CL_TASK NULL |
| \$7: DUMP_STORAGE | Storage | CS_CL_STORAGE NULL |
| \$8: DUMP_PROGRAM | Program | CS_CL_PROGRAM NULL |
| \$9: DUMP_TERMINAL | Terminal | CS_CL_TERMINAL NULL |
| \$10: DUMP_TABLES | Tables | CS_CL_TABLES NULL |
| \$11: DUMP_DCT | Dct | CS_CL_DCT NULL |
| \$12: DUMP_FCT | Fct | CS_CL_FCT NULL |
| \$13: DUMP_PCT | Pct | CS_CL_PCT NULL |
| \$14: DUMP_PPT | Ppt | CS_CL_PPT NULL |
| \$15: DUMP_SIT | Sit | CS_CL_SIT NULL |
| \$16: DUMP_TCT | Tct | CS_CL_TCT NULL |
| \$17: DUMP_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$18: DUMP_RESP | Resp | CS_CL_RESP NULL |
| \$19: DUMP_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_ENDBR | Endbr Statement (CICS) | ENDBR\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
|-------------------------|------------------------|--|
| \$1: ENDBR_NOT_USED | not-used | NULL |
| \$2: ENDBR_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: ENDBR_REQID | Reqid | CS_CL_REQID NULL |
| \$4: ENDBR_SYSID | Sysid | CS_CL_SYSID NULL |
| \$5: ENDBR_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$6: ENDBR_RESP | Resp | CS_CL_RESP NULL |
| \$7: ENDBR_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_FORMATTIME | Formattime Statement (CICS) | FORMATTIME\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
|---------------------------|-----------------------------|---|
| \$1: FMETIME_NOT_USED | not-used | NULL |
| \$2: FMETIME_ABSTIME | Abstime | CS_CL_ABSTIME |
| \$3: FMETIME_DATE | Date | CS_CL_DATE NULL |
| \$4: FMETIME_FULldATE | Fulldate | CS_CL_FULldATE NULL |
| \$5: FMETIME_DATEFORM | Dateform | CS_CL_DATEFORM NULL |
| \$6: FMETIME_DATESEP | Datesep | CS_CL_DATESEP CS_CL_DATESEP NULL |
| \$7: FMETIME_DAYCOUNT | Daycount | CS_CL_DAYCOUNT NULL |
| \$8: FMETIME_DAYOFMONTH | Dayofmonth | CS_CL_DAYOFMONTH NULL |
| \$9: FMETIME_DAYOFWEEK | Dayofweek | CS_CL_DAYOFWEEK NULL |
| \$10: FMETIME_DDMMYY | Ddmmyy | CS_CL_DDMMYY NULL |
| \$11: FMETIME_DDMMYYYY | Ddmmyyyy | CS_CL_DDMMYYYY NULL |
| \$12: FMETIME_MMDDYY | Mmddyy | CS_CL_MMDDYY NULL |
| \$13: FMETIME_MMDDYYYY | Mmddyyyy | CS_CL_MMDDYYYY NULL |
| \$14: FMETIME_MONTHOFYEAR | Monthofyear | CS_CL_MONTHOFYEAR NULL |
| \$15: FMETIME_TIME | Time | CS_CL_TIME NULL |
| \$16: FMETIME_TIMESEP | Timesep | CS_CL_TIMESEP CS_CL_TIMESEP NULL |
| \$17: FMETIME_YEAR | Year | CS_CL_YEAR NULL |
| \$18: FMETIME_YYDDD | Yydd | CS_CL_YYDDD NULL |
| \$19: FMETIME_YYDDMM | Yyddmm | CS_CL_YYDDMM NULL |
| \$20: FMETIME_YYMMDD | Yymmdd | CS_CL_YYMMDD NULL |
| \$21: FMETIME_YYYYDDD | Yyyyddd | CS_CL_YYYYDDD NULL |
| \$22: FMETIME_YYYYDDMM | Yyyyddmm | CS_CL_YYYYDDMM NULL |
| \$23: FMETIME_YYYYMMDD | Yyyymmdd | CS_CL_YYYYMMDD NULL |
| \$24: FMETIME_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$25: FMETIME_RESP | Resp | CS_CL_RESP NULL |
| \$26: FMETIME_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|------------------------|---------------------------|--|
| CS_STMT_FREEMAIN | Freemain Statement (CICS) | FREEMAIN\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: FREEMAIN_NOT_USED | not-used | NULL |
| \$2: FREEMAIN_DATA | Data | CS_CL_DATA CS_CL_DATAPOINTER |
| \$3: FREEMAIN_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: FREEMAIN_RESP | Resp | CS_CL_RESP NULL |
| \$5: FREEMAIN_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|------------------------|--------------------------|---|
| CS_STMT_GETMAIN | Getmain Statement (CICS) | GETMAIN\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: GETMAIN_NOT_USED | not-used | NULL |
| \$2: GETMAIN_SET | Set | CS_CL_SET |
| \$3: GETMAIN_LENGTH | Length | CS_CL_FLENGTH CS_CL_LENGTH |
| \$4: GETMAIN_BELOW | Below | CS_CL_BELOW NULL |
| \$5: GETMAIN_INITIMG | Initimg | CS_CL_INITIMG NULL |
| \$6: GETMAIN_SHARED | Shared | CS_CL_SHARED NULL |
| \$7: GETMAIN_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$8: GETMAIN_DATAKEY | Datakey | CS_CL_USERDATAKEY CS_CL_CICSDATAKEY NULL |
| \$9: GETMAIN_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$10: GETMAIN_RESP | Resp | CS_CL_RESP NULL |
| \$11: GETMAIN_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|----------------------|-------------------------------|--|
| CS_STMT_HANDLE_ABEND | Handle Abend Statement (CICS) | HANDLE ABEND\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: HNDAB_NOT_USED | not-used | NULL |
| \$2: HNDAB_OPTION | Option | CS_CL_CANCEL CS_CL_PROGRAM CS_CL_LABEL CS_CL_RESET NULL |
| \$3: HNDAB_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: HNDAB_RESP | Resp | CS_CL_RESP NULL |
| \$5: HNDAB_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|------------------------------|-----------------------------|--|
| CS_STMT_HANDLE_AID | Handle Aid Statement (CICS) | HANDLE AID\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: HNDAID_NOT_USED | not-used | NULL |
| \$2: HNDAID_AID_HANDLER_LIST | Aid Handler List | aid-handler-list |
| \$3: HNDAID_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: HNDAID_RESP | Resp | CS_CL_RESP NULL |
| \$5: HNDAID_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-------------------------------------|-----------------------------------|---|
| CS_STMT_HANDLE_CONDITION | Handle Condition Statement (CICS) | HANDLE CONDITION\$1{\$2?\n\$2}{\$3?\n\$3} |
| \$1: HNDCOND_NOT_USED | not-used | NULL |
| \$2: HNDCOND_CONDITION_HANDLER_LIST | Condition Handler List | condition-handler-list |
| \$3: HNDCOND_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: HNDCOND_RESP | Resp | CS_CL_RESP NULL |
| \$5: HNDCOND_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-------------------------------------|-----------------------------------|---|
| CS_STMT_IGNORE_CONDITION | Ignore Condition Statement (CICS) | IGNORE CONDITION\$1{\$2?\n\$2}{\$3?\n\$3} |
| \$1: IGNCOND_NOT_USED | not-used | NULL |
| \$2: IGNCOND_CONDITION_HANDLER_LIST | Condition Handler List | condition-handler-list |
| \$3: IGNCOND_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$4: IGNCOND_RESP | Resp | CS_CL_RESP NULL |
| \$5: IGNCOND_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-----------------------|-----------------------|---|
| CS_STMT_LINK | Link Statement (CICS) | LINK\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
| \$1: LNK_NOT_USED | not-used | NULL |
| \$2: LNK_PROGRAM | Program | CS_CL_PROGRAM |
| \$3: LNK_COMMAREA | Commarea | CS_CL_COMMAREA NULL |
| \$4: LNK_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: LNK_DATALENGTH | Datalength | CS_CL_DATALENGTH NULL |
| \$6: LNK_INPUTMSG | Inputmsg | CS_CL_INPUTMSG NULL |
| \$7: LNK_INPUTMSGLEN | Inputmsglen | CS_CL_INPUTMSGLEN NULL |
| \$8: LNK_SYSID | Sysid | CS_CL_SYSID NULL |
| \$9: LNK_SYNCONRETURN | Synconreturn | CS_CL_SYNCONRETURN NULL |
| \$10: LNK_TRANSID | Transid | CS_CL_TRANSID NULL |
| \$11: LNK_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$12: LNK_RESP | Resp | CS_CL_RESP NULL |
| \$13: LNK_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------|------------------------------|---|
| CS_STMT_PUSH_HANDLE | Push Handle Statement (CICS) | PUSH HANDLE\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: PUSH_NOT_USED | not-used | NULL |
| \$2: PUSH_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$3: PUSH_RESP | Resp | CS_CL_RESP NULL |
| \$4: PUSH_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-------------------------------------|-----------------------|--|
| CS_STMT_READ | Read Statement (CICS) | READ\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5? |
| \$1: READ_NOT_USED | not-used | NULL |
| \$2: READ_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: READ_LOCKMODE | Lockmode | CS_CL_UNCOMMITTED CS_CL_CONSISTENT CS_CL_REPEATABLE NULL |
| \$4: READ_UPDATE | Update | CS_CL_UPDATE NULL |
| \$5: READ_TOKEN | Token | CS_CL_TOKEN NULL |
| \$6: READ_INTO_SET | Into/Set | CS_CL_INTO CS_CL_SET |
| \$7: READ_LENGTH | Length | CS_CL_LENGTH NULL |
| \$8: READ_RIDFLD | Ridfld | CS_CL_RIDFLD |
| \$9: READ_KEYLENGTH | Keylength | CS_CL_KEYLENGTH NULL |
| \$10: READ_GENERIC | Generic | CS_CL_GENERIC NULL |
| \$11: READ_SYSID | Sysid | CS_CL_SYSID NULL |
| \$12: READ_RBA_RRN_DEBKEY_DEBREC | Rba/Rrn/Debkey/Debrec | CS_CL_RBA CS_CL_RRN CS_CL_DEBKEY CS_CL_DEBREC NULL |
| \$13: READ_EQUAL_GTEQ | Equal/Gteq | CS_CL_EQUAL CS_CL_GTEQ NULL |
| \$14: READ_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$15: READ_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$16: READ_RESP | Resp | CS_CL_RESP NULL |
| \$17: READ_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_READNEXT | Readnext Statement (CICS) | READNEXT\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
|--------------------------|---------------------------|--|
| \$1: RDNEXT_NOT_USED | not-used | NULL |
| \$2: RDNEXT_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: RDNEXT_INT0_SET | Into/Set | CS_CL_INT0 CS_CL_SET |
| \$4: RDNEXT_LOCKMODE | Lockmode | CS_CL_UNCOMMITTED CS_CL_CONSISTENT CS_CL_REPEATABLE NULL |
| \$5: RDNEXT_UPDATE | Update | CS_CL_UPDATE NULL |
| \$6: RDNEXT_TOKEN | Token | CS_CL_TOKEN NULL |
| \$7: RDNEXT_LENGTH | Length | CS_CL_LENGTH NULL |
| \$8: RDNEXT_RIDFLD | Ridfld | CS_CL_RIDFLD |
| \$9: RDNEXT_KEYLENGTH | Keylength | CS_CL_KEYLENGTH NULL |
| \$10: RDNEXT_REQID | Reqid | CS_CL_REQID NULL |
| \$11: RDNEXT_SYSID | Sysid | CS_CL_SYSID NULL |
| \$12: RDNEXT_RBA_RRN | Rba/Rrn | CS_CL_RBA CS_CL_RRN NULL |
| \$13: RDNEXT_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$14: RDNEXT_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$15: RDNEXT_RESP | Resp | CS_CL_RESP NULL |
| \$16: RDNEXT_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_READPREV | Readprev Statement (CICS) | READPREV\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
|--------------------------|---------------------------|--|
| \$1: RDPREV_NOT_USED | not-used | NULL |
| \$2: RDPREV_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: RDPREV_INT0_SET | Into/Set | CS_CL_INT0 CS_CL_SET |
| \$4: RDPREV_LOCKMODE | Lockmode | CS_CL_UNCOMMITTED CS_CL_CONSISTENT CS_CL_REPEATABLE NULL |
| \$5: RDPREV_UPDATE | Update | CS_CL_UPDATE NULL |
| \$6: RDPREV_TOKEN | Token | CS_CL_TOKEN NULL |
| \$7: RDPREV_LENGTH | Length | CS_CL_LENGTH NULL |
| \$8: RDPREV_RIDFLD | Ridfld | CS_CL_RIDFLD |
| \$9: RDPREV_KEYLENGTH | Keylength | CS_CL_KEYLENGTH NULL |
| \$10: RDPREV_REQID | Reqid | CS_CL_REQID NULL |
| \$11: RDPREV_SYSID | Sysid | CS_CL_SYSID NULL |
| \$12: RDPREV_RBA_RRN | Rba/Rrn | CS_CL_RBA CS_CL_RRN NULL |
| \$13: RDPREV_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$14: RDPREV_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$15: RDPREV_RESP | Resp | CS_CL_RESP NULL |
| \$16: RDPREV_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|----------------------|---------------------------|--|
| CS_STMT_READQ_TS | Readq Ts Statement (CICS) | READQ TS\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: RDQTS_NOT_USED | not-used | NULL |
| \$2: RDQTS_QUEUE | Queue | CS_CL_QUEUE CS_CL_QNAME |
| \$3: RDQTS_INT0_SET | Into/Set | CS_CL_INT0 CS_CL_SET |
| \$4: RDQTS_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: RDQTS_NUMITEMS | Numitems | CS_CL_NUMITEMS NULL |
| \$6: RDQTS_NEXT_ITEM | Next/Item | CS_CL_NEXT CS_CL_ITEM NULL |
| \$7: RDQTS_SYSID | Sysid | CS_CL_SYSID NULL |
| \$8: RDQTS_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$9: RDQTS_RESP | Resp | CS_CL_RESP NULL |
| \$10: RDQTS_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------|--------------------------|--|
| CS_STMT_RECEIVE | Receive Statement (CICS) | RECEIVE\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: RCV_NOT_USED | not-used | NULL |
| \$2: RCV_INT0_SET | Into/Set | CS_CL_INT0 CS_CL_SET NULL |
| \$3: RCV_LENGTH | Length | CS_CL_LENGTH CS_CL_FLENGTH |
| \$4: RCV_PSEUDOBIN | Pseudobin | CS_CL_PSEUDOBIN NULL |
| \$5: RCV_MAXLENGTH | Maxlength | CS_CL_MAXLENGTH CS_CL_MAXFLENGTH NULL |
| \$6: RCV_NOTRUNCATE | Nottruncate | CS_CL_NOTRUNCATE NULL |
| \$7: RCV_ASIS | Asis | CS_CL_ASIS NULL |
| \$8: RCV_BUFFER | Buffer | CS_CL_BUFFER NULL |
| \$9: RCV_CONVID | Convid | CS_CL_CONVID NULL |
| \$10: RCV_LEAVEKB | Leavekb | CS_CL_LEAVEKB NULL |
| \$11: RCV_PASSBK | Passbk | CS_CL_PASSBK |
| \$12: RCV_SESSION | Session | CS_CL_SESSION NULL |
| \$13: RCV_STATE | State | CS_CL_STATE NULL |
| \$14: RCV_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$15: RCV_RESP | Resp | CS_CL_RESP NULL |
| \$16: RCV_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|------------------------|------------------------------|--|
| CS_STMT_RECEIVE_MAP | Receive Map Statement (CICS) | RECEIVE{\$1?\n\$1}{\$2?\n\$2}{\$3?\n\$3}{} |
| \$1: RCVMAP_MAP | Map | CS_CL_MAP |
| \$2: RCVMAP_MAPPINGDEV | Mappingdev | CS_CL_MAPPINGDEV |
| \$3: RCVMAP_LENGTH | Length | CS_CL_LENGTH NULL |
| \$4: RCVMAP_MAPSET | Mapset | CS_CL_MAPSET NULL |
| \$5: RCVMAP_INTTO_SET | Into/Set | CS_CL_INTTO CS_CL_SET NULL |
| \$6: RCVMAP_TERMINAL | Terminal | CS_CL_TERMINAL NULL |
| \$7: RCVMAP_ASIS | Asis | CS_CL_ASIS NULL |
| \$8: RCVMAP_INPARTN | Inpartn | CS_CL_INPARTN NULL |
| \$9: RCVMAP_FROM | From | CS_CL_FROM NULL |
| \$10: RCVMAP_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$11: RCVMAP_RESP | Resp | CS_CL_RESP NULL |
| \$12: RCVMAP_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------|---------------------------|--|
| CS_STMT_RETRIEVE | Retrieve Statement (CICS) | RETRIEVE\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{} |
| \$1: RETR_NOT_USED | not-used | NULL |
| \$2: RETR_INTTO_SET | Into/Set | CS_CL_INTTO CS_CL_SET |
| \$3: RETR_LENGTH | Length | CS_CL_LENGTH NULL |
| \$4: RETR_RTRANSID | Rtransid | CS_CL_RTRANSID NULL |
| \$5: RETR_RTERMID | Rtermid | CS_CL_RTERMID NULL |
| \$6: RETR_QUEUE | Queue | CS_CL_QUEUE NULL |
| \$7: RETR_WAIT | Wait | CS_CL_WAIT NULL |
| \$8: RETR_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$9: RETR_RESP | Resp | CS_CL_RESP NULL |
| \$10: RETR_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-----------------------|-------------------------|--|
| CS_STMT_RETURN | Return Statement (CICS) | RETURN\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{} |
| \$1: RETN_NOT_USED | not-used | NULL |
| \$2: RETN_TRANSID | Transid | CS_CL_TRANSID NULL |
| \$3: RETN_COMMAREA | Commarea | CS_CL_COMMAREA NULL |
| \$4: RETN_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: RETN_IMMEDIATE | Immediate | CS_CL_IMMEDIATE NULL |
| \$6: RETN_INPUTMSG | Inputmsg | CS_CL_INPUTMSG NULL |
| \$7: RETN_INPUTMSGLEN | Inputmsglen | CS_CL_INPUTMSGLEN NULL |
| \$8: RETN_ENDACTIVITY | Endactivity | CS_CL_ENDACTIVITY NULL |
| \$9: RETN_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$10: RETN_RESP | Resp | CS_CL_RESP NULL |
| \$11: RETN_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|------------------------|--------------------------|---|
| CS_STMT_REWRITE | Rewrite Statement (CICS) | REWRITE\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: REWR_NOT_USED | not-used | NULL |
| \$2: REWR_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: REWR_TOKEN | Token | CS_CL_TOKEN NULL |
| \$4: REWR_FROM | From | CS_CL_FROM |
| \$5: REWR_LENGTH | Length | CS_CL_LENGTH NULL |
| \$6: REWR_SYSID | Sysid | CS_CL_SYSID NULL |
| \$7: REWR_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$8: REWR_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$9: REWR_RESP | Resp | CS_CL_RESP NULL |
| \$10: REWR_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------------------|-----------------------|---|
| CS_STMT_SEND | Send Statement (CICS) | SEND\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5? |
| \$1: SND_NOT_USED | not-used | NULL |
| \$2: SND_CONVID | Convid | CS_CL_CONVID NULL |
| \$3: SND_CTLCHAR | Ctlchar | CS_CL_CTLCHAR NULL |
| \$4: SND_DEST | Dest | CS_CL_DEST NULL |
| \$5: SND_FROM | From | CS_CL_FROM NULL |
| \$6: SND_LINEADDR | Lineaddr | CS_CL_LINEADDR NULL |
| \$7: SND_LEAVEKB | Leavekb | CS_CL_LEAVEKB NULL |
| \$8: SND_PSEUDOBIN | Pseudobin | CS_CL_PSEUDOBIN NULL |
| \$9: SND_ASIS | Asis | CS_CL_ASIS NULL |
| \$10: SND_SESSION | Session | CS_CL_SESSION NULL |
| \$11: SND_DEF_ALT | Def/Alt | CS_CL_DEFAULT CS_CL_ALTERNATE NULL |
| \$12: SND_LDC_FMH | Ldc/Fmh | CS_CL_LDC CS_CL_FMH NULL |
| \$13: SND_INVITE_LAST | Invite/Last | CS_CL_INVITE CS_CL_LAST NULL |
| \$14: SND_CNOTCOMPL_DEFRESP | Cnotcompl/Defresp | CS_CL_CNOTCOMPL CS_CL_DEFRESP NULL |
| \$15: SND_CONFIRM_WAIT | Confirm/Wait | CS_CL_CONFIRM CS_CL_WAIT NULL |
| \$16: SND_ATTACHID | Attachid | CS_CL_ATTACHID NULL |
| \$17: SND_STATE | State | CS_CL_STATE NULL |
| \$18: SND_ERASE_STRFIELD | Erase/Strfield | CS_CL_ERASE CS_CL_STRFIELD NULL |
| \$19: SND_LENGTH | Length | CS_CL_LENGTH CS_CL_FLENGTH NULL |
| \$20: SND_PASSBK_CBUFF | Passbk/Cbuff | CS_CL_PASSBK CS_CL_CBUFF |
| \$21: SND_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$22: SND_RESP | Resp | CS_CL_RESP NULL |
| \$23: SND_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------------|---------------------------|--|
| CS_STMT_SEND_MAP | Send Map Statement (CICS) | SEND{\$1?\n\$1}{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: SNDMAP_MAP | Map | CS_CL_MAP |
| \$2: SNDMAP_MAPPINGDEV | Mappingdev | CS_CL_MAPPINGDEV |
| \$3: SNDMAP_SET | Set | CS_CL_SET |
| \$4: SNDMAP_MAPSET | Mapset | CS_CL_MAPSET NULL |
| \$5: SNDMAP_FROM_MAPONLY | From/Maponly | CS_CL_FROM CS_CL_MAPONLY NULL |
| \$6: SNDMAP_DATAONLY | Dataonly | CS_CL_DATAONLY NULL |
| \$7: SNDMAP_LENGTH | Length | CS_CL_LENGTH NULL |
| \$8: SNDMAP_CURSOR | Cursor | CS_CL_CURSOR CS_CL_CURSOR NULL |
| \$9: SNDMAP_FORMFEED | Formfeed | CS_CL_FORMFEED NULL |
| \$10: SNDMAP_ERASE | Erase | CS_CL_ERASE CS_CL_ERASEAUP NULL |
| \$11: SNDMAP_PRINT | Print | CS_CL_PRINT NULL |
| \$12: SNDMAP_FREEKB | Freekb | CS_CL_FREEKB NULL |
| \$13: SNDMAP_ALARM | Alarm | CS_CL_ALARM NULL |
| \$14: SNDMAP_FRSET | Frset | CS_CL_FRSET NULL |
| \$15: SNDMAP_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$16: SNDMAP_RESP | Resp | CS_CL_RESP NULL |
| \$17: SNDMAP_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------------------|----------------------------|--|
| CS_STMT_SEND_TEXT | Send Text Statement (CICS) | SEND TEXT\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
| \$1: SNDTXT_NOT_USED | not-used | NULL |
| \$2: SNDTXT_NOEDIT | Noedit | CS_CL_NOEDIT |
| \$3: SNDTXT_FROM | From | CS_CL_FROM |
| \$4: SNDTXT_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: SNDTXT_ERASE | Erase | CS_CL_ERASE NULL |
| \$6: SNDTXT_DEF_ALT | Def/Alt | CS_CL_DEFAULT CS_CL_ALTERNATE NULL |
| \$7: SNDTXT_PRINT | Print | CS_CL_PRINT NULL |
| \$8: SNDTXT_FREEKB | Freekb | CS_CL_FREEKB NULL |
| \$9: SNDTXT_ALARM | Alarm | CS_CL_ALARM NULL |
| \$10: SNDTXT_MAPPED | Mapped | CS_CL_MAPPED |
| \$11: SNDTXT_OUTPARTN | Outpartn | CS_CL_OUTPARTN NULL |
| \$12: SNDTXT_TERMINAL_PAGING | Terminal/Paging | CS_CL_TERMINAL CS_CL_PAGING NULL |
| \$13: SNDTXT_WAIT | Wait | CS_CL_WAIT NULL |
| \$14: SNDTXT_LAST | Last | CS_CL_LAST NULL |
| \$15: SNDTXT_REQID | Reqid | CS_CL_REQID NULL |
| \$16: SNDTXT_HONEOM_LNN | Honeom/Lnn | CS_CL_HONEOM CS_CL_L40 CS_CL_L64 CS_CL_L80 NULL |
| \$17: SNDTXT_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$18: SNDTXT_RESP | Resp | CS_CL_RESP NULL |
| \$19: SNDTXT_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|---------------------------|------------------------|--|
| CS_STMT_START | Start Statement (CICS) | START\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5?\n\$5} |
| \$1: START_NOT_USED | not-used | NULL |
| \$2: START_TRANSID | Transid | CS_CL_TRANSID |
| \$3: START_INTERVAL_TIME | Interval/Time | CS_CL_INTERVAL CS_CL_TIME NULL |
| \$4: START_MODE | Mode | CS_CL_AFTER CS_CL_AT CS_CL_FOR CS_CL_UNTIL NULL |
| \$5: START_HOURS | Hours | CS_CL_HOURS NULL |
| \$6: START_MINUTES | Minutes | CS_CL_MINUTES NULL |
| \$7: START_SECONDS | Seconds | CS_CL_SECONDS NULL |
| \$8: START_REQID | Reqid | CS_CL_REQID NULL |
| \$9: START_FROM | From | CS_CL_FROM NULL |
| \$10: START_LENGTH | Length | CS_CL_LENGTH NULL |
| \$11: START_FMH | Fmh | CS_CL_FMH NULL |
| \$12: START_TERMID_USERID | Termid/Userid | CS_CL_TERMID CS_CL_USERID NULL |
| \$13: START_SYSID | Sysid | CS_CL_SYSID NULL |
| \$14: START_RTRANSID | Rtransid | CS_CL_RTRANSID NULL |
| \$15: START_RTERMID | Rtermid | CS_CL_RTERMID NULL |
| \$16: START_QUEUE | Queue | CS_CL_QUEUE NULL |
| \$17: START_NOCHECK | Nocheck | CS_CL_NOCHECK NULL |
| \$18: START_PROTECT | Protect | CS_CL_PROTECT NULL |
| \$19: START_ATTACH | Attach | CS_CL_ATTACH NULL |
| \$20: START_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$21: START_RESP | Resp | CS_CL_RESP NULL |
| \$22: START_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_STARTBR | Startbr Statement (CICS) | STARTBR\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
|---------------------------------------|--------------------------|---|
| \$1: STARTBR_NOT_USED | not-used | NULL |
| \$2: STARTBR_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: STARTBR_RIDFLD | Ridfld | CS_CL_RIDFLD |
| \$4: STARTBR_KEYLENGTH | Keylength | CS_CL_KEYLENGTH NULL |
| \$5: STARTBR_GENERIC | Generic | CS_CL_GENERIC NULL |
| \$6: STARTBR_REQID | Reqid | CS_CL_REQID NULL |
| \$7: STARTBR_SYSID | Sysid | CS_CL_SYSID NULL |
| \$8: STARTBR_RBA_RRN_DEBKEY_DEBREC | Rba/Rrn/Debkey/Debrec | CS_CL_RBA CS_CL_RRN CS_CL_DEBKEY CS_CL_DEBREC NULL |
| \$9: STARTBR_GTEQ_EQUAL | Gteq/Equal | CS_CL_GTEQ CS_CL_EQUAL NULL |
| \$10: STARTBR_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$11: STARTBR_RESP | Resp | CS_CL_RESP NULL |
| \$12: STARTBR_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_SYNCPOINT | Syncpoint Statement (CICS) | SYNCPOINT\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
|---------------------|----------------------------|---|
| \$1: SYNCP_NOT_USED | not-used | NULL |
| \$2: SYNCP_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$3: SYNCP_RESP | Resp | CS_CL_RESP NULL |
| \$4: SYNCP_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| CS_STMT_UNLOCK | Unlock Statement (CICS) | UNLOCK\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4} |
|--------------------------|-------------------------|--|
| \$1: UNLOCK_NOT_USED | not-used | NULL |
| \$2: UNLOCK_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: UNLOCK_TOKEN | Token | CS_CL_TOKEN NULL |
| \$4: UNLOCK_SYSID | Sysid | CS_CL_SYSID NULL |
| \$5: UNLOCK_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$6: UNLOCK_RESP | Resp | CS_CL_RESP NULL |
| \$7: UNLOCK_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-----------------------|------------------------|---|
| CS_STMT_WRITE | Write Statement (CICS) | WRITE\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\n\$4}{\n\$4} |
| \$1: WRT_NOT_USED | not-used | NULL |
| \$2: WRT_FILE_DATASET | File/Dataset | CS_CL_FILE CS_CL_DATASET |
| \$3: WRT_MASSINSERT | Massinsert | CS_CL_MASSINSERT NULL |
| \$4: WRT_FROM | From | CS_CL_FROM |
| \$5: WRT_LENGTH | Length | CS_CL_LENGTH NULL |
| \$6: WRT_RIDFLD | Ridfld | CS_CL_RIDFLD |
| \$7: WRT_KEYLENGTH | Keylength | CS_CL_KEYLENGTH NULL |
| \$8: WRT_SYSID | Sysid | CS_CL_SYSID NULL |
| \$9: WRT_RBA_RRN | Rba/Rrn | CS_CL_RBA CS_CL_RRN NULL |
| \$10: WRT_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$11: WRT_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$12: WRT_RESP | Resp | CS_CL_RESP NULL |
| \$13: WRT_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|----------------------|----------------------------|---|
| CS_STMT_WRITEQ_TD | Writeq Td Statement (CICS) | WRITEQ TD\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\n\$4}{\n\$4} |
| \$1: WRTQTD_NOT_USED | not-used | NULL |
| \$2: WRTQTD_QUEUE | Queue | CS_CL_QUEUE |
| \$3: WRTQTD_FROM | From | CS_CL_FROM |
| \$4: WRTQTD_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: WRTQTD_SYSID | Sysid | CS_CL_SYSID NULL |
| \$6: WRTQTD_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$7: WRTQTD_RESP | Resp | CS_CL_RESP NULL |
| \$8: WRTQTD_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|--------------------------|----------------------------|---|
| CS_STMT_WRITEQ_TS | Writeq Ts Statement (CICS) | WRITEQ TS\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\n\$4}{\n\$4} |
| \$1: WRTS_NOT_USED | not-used | NULL |
| \$2: WRTS_QUEUE | Queue | CS_CL_QUEUE CS_CL_QNAME |
| \$3: WRTS_FROM | From | CS_CL_FROM |
| \$4: WRTS_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: WRTS_NUMITEMS | Numitems | CS_CL_NUMITEMS NULL |
| \$6: WRTS_ITEM | Item | CS_CL_ITEM NULL |
| \$7: WRTS_REWRITE | Rewrite | CS_CL_REWRITE NULL |
| \$8: WRTS_SYSID | Sysid | CS_CL_SYSID NULL |
| \$9: WRTS_AUXILIARY_MAIN | Auxiliary/Main | CS_CL_AUXILIARY CS_CL_MAIN NULL |
| \$10: WRTS_NOSUSPEND | Nosuspend | CS_CL_NOSUSPEND NULL |
| \$11: WRTS_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$12: WRTS_RESP | Resp | CS_CL_RESP NULL |
| \$13: WRTS_RESP2 | Resp2 | CS_CL_RESP2 NULL |

| | | |
|-----------------------|-----------------------|---|
| CS_STMT_XCTL | Xctl Statement (CICS) | XCTL\$1{\$2?\n\$2}{\$3?\n\$3}{\$4?\n\$4}{\$5? |
| \$1: XCTL_NOT_USED | not-used | NULL |
| \$2: XCTL_PROGRAM | Program | CS_CL_PROGRAM |
| \$3: XCTL_COMMAREA | Commarea | CS_CL_COMMAREA NULL |
| \$4: XCTL_LENGTH | Length | CS_CL_LENGTH NULL |
| \$5: XCTL_INPUTMSG | Inputmsg | CS_CL_INPUTMSG NULL |
| \$6: XCTL_INPUTMSGLEN | Inputmsglen | CS_CL_INPUTMSGLEN NULL |
| \$7: XCTL_NOHANDLE | Nohandle | CS_CL_NOHANDLE NULL |
| \$8: XCTL_RESP | Resp | CS_CL_RESP NULL |
| \$9: XCTL_RESP2 | Resp2 | CS_CL_RESP2 NULL |

10.10.2 Options

| | | |
|------------------------|----------------------|-----------------------|
| CS_CL_ABCODE | Abcode Clause (CICS) | ABCODE{\$1?\$p2(\$1)} |
| \$1: CS_CL_ABCODE__ARG | Option value | cs-name |

| | | |
|-------------------------|-----------------------|------------------------|
| CS_CL_ABSTIME | Abstime Clause (CICS) | ABSTIME{\$1?\$p2(\$1)} |
| \$1: CS_CL_ABSTIME__ARG | Option value | cs-data-area |

| | | |
|----------------------|--------------------|---------------------|
| CS_CL_ACEE | Acee Clause (CICS) | ACEE{\$1?\$p2(\$1)} |
| \$1: CS_CL_ACEE__ARG | Option value | cs-ptr-ref |

| | | |
|-------------|---------------------|-------|
| CS_CL_AFTER | After Clause (CICS) | AFTER |
|-------------|---------------------|-------|

| | | |
|-------------|---------------------|-------|
| CS_CL_ALARM | Alarm Clause (CICS) | ALARM |
|-------------|---------------------|-------|

| | | |
|-----------------|-------------------------|-----------|
| CS_CL_ALTERNATE | Alternate Clause (CICS) | ALTERNATE |
|-----------------|-------------------------|-----------|

| | | |
|------------|--------------------|------|
| CS_CL_ASIS | Asis Clause (CICS) | ASIS |
|------------|--------------------|------|

| | | |
|--------------------|------------------|-------------------|
| CS_CL_AT | At Clause (CICS) | AT{\$1?\$p2(\$1)} |
| \$1: CS_CL_AT__ARG | Option value | cs-name NULL |

| | | |
|--------------|----------------------|--------|
| CS_CL_ATTACH | Attach Clause (CICS) | ATTACH |
|--------------|----------------------|--------|

| | | |
|--------------------------|------------------------|-------------------------|
| CS_CL_ATTACHID | Attachid Clause (CICS) | ATTACHID{\$1?\$p2(\$1)} |
| \$1: CS_CL_ATTACHID__ARG | Option value | cs-name |

| | | |
|-----------------|-------------------------|-----------|
| CS_CL_AUXILIARY | Auxiliary Clause (CICS) | AUXILIARY |
|-----------------|-------------------------|-----------|

| | | |
|-------------------------|---------------------------|-------------------------|
| CS_CL_BELOW | Below Clause (CICS) | BELOW |
| CS_CL_BUFFER | Buffer Clause (CICS) | BUFFER |
| CS_CL_CANCEL | Cancel Clause (CICS) | CANCEL |
| CS_CL_CBUFF | Cbuff Clause (CICS) | CBUFF |
| CS_CL_CICSATAKEY | Cicsdatakey Clause (CICS) | CICSATAKEY |
| CS_CL_CNOTCOMPL | Cnotcompl Clause (CICS) | CNOTCOMPL |
| CS_CL_COMMAREA | Commarea Clause (CICS) | COMMAREA{\$1?\$p2(\$1)} |
| \$1: CS_CL_COMMAREA_ARG | Option value | cs-arg-any |
| CS_CL_COMPLETE | Complete Clause (CICS) | COMPLETE |
| CS_CL_CONFIRM | Confirm Clause (CICS) | CONFIRM |
| CS_CL_CONSISTENT | Consistent Clause (CICS) | CONSISTENT |
| CS_CL_CONVID | Convid Clause (CICS) | CONVID{\$1?\$p2(\$1)} |
| \$1: CS_CL_CONVID_ARG | Option value | cs-arg-any |
| CS_CL_CTLCHAR | Ctlchar Clause (CICS) | CTLCHAR{\$1?\$p2(\$1)} |
| \$1: CS_CL_CTLCHAR_ARG | Option value | cs-data-value |
| CS_CL_CURSOR | Cursor Clause (CICS) | CURSOR{\$1?\$p2(\$1)} |
| \$1: CS_CL_CURSOR_ARG | Option value | cs-data-value NULL |
| CS_CL_CWA | Cwa Clause (CICS) | CWA{\$1?\$p2(\$1)} |
| \$1: CS_CL_CWA_ARG | Option value | cs-ptr-ref |
| CS_CL_DATA | Data Clause (CICS) | DATA{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATA_ARG | Option value | cs-data-area |

| | | |
|-----------------------------|---------------------------|----------------------------|
| CS_CL_DATALENGTH | Datalength Clause (CICS) | DATALENGTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATALENGTH__ARG | Option value | cs-data-value |
| CS_CL_DATAONLY | Dataonly Clause (CICS) | DATAONLY |
| CS_CL_DATAPOINTER | Datapointer Clause (CICS) | DATAPOINTER{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATAPOINTER__ARG | Option value | cs-ptr-value |
| CS_CL_DATASET | Dataset Clause (CICS) | DATASET{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATASET__ARG | Option value | cs-filename |
| CS_CL_DATE | Date Clause (CICS) | DATE{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATE__ARG | Option value | cs-data-area |
| CS_CL_DATEFORM | Dateform Clause (CICS) | DATEFORM{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATEFORM__ARG | Option value | cs-data-area |
| CS_CL_DATESEP | Datesep Clause (CICS) | DATESEP{\$1?\$p2(\$1)} |
| \$1: CS_CL_DATESEP__ARG | Option value | cs-data-value NULL |
| CS_CL_DAYCOUNT | Daycount Clause (CICS) | DAYCOUNT{\$1?\$p2(\$1)} |
| \$1: CS_CL_DAYCOUNT__ARG | Option value | cs-data-area |
| CS_CL_DAYOFMONTH | Dayofmonth Clause (CICS) | DAYOFMONTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_DAYOFMONTH__ARG | Option value | cs-data-area |
| CS_CL_DAYOFWEEK | Dayofweek Clause (CICS) | DAYOFWEEK{\$1?\$p2(\$1)} |
| \$1: CS_CL_DAYOFWEEK__ARG | Option value | cs-data-area |
| CS_CL_DCT | Dct Clause (CICS) | DCT |
| CS_CL_DDMMYY | Ddmmyy Clause (CICS) | DDMMYY{\$1?\$p2(\$1)} |
| \$1: CS_CL_DDMMYY__ARG | Option value | cs-data-area |
| CS_CL_DDMMYYYY | Ddmmyyyy Clause (CICS) | DDMMYYYY{\$1?\$p2(\$1)} |
| \$1: CS_CL_DDMMYYYY__ARG | Option value | cs-data-area |
| CS_CL_DEBKEY | Debkey Clause (CICS) | DEBKEY |
| CS_CL_DEBREC | Debrec Clause (CICS) | DEBREC |

| | | |
|---------------------------|---------------------------|-------------------------|
| CS_CL_DEFAULT | Default Clause (CICS) | DEFAULT |
| CS_CL_DEFRESP | Defresp Clause (CICS) | DEFRESP |
| CS_CL_DEST | Dest Clause (CICS) | DEST{\$1?\$p2(\$1)} |
| \$1: CS_CL_DEST__ARG | Option value | cs-name |
| CS_CL_DUMP CODE | Dumpcode Clause (CICS) | DUMPCODE{\$1?\$p2(\$1)} |
| \$1: CS_CL_DUMP CODE__ARG | Option value | cs-name |
| CS_CL_EIB | Eib Clause (CICS) | EIB{\$1?\$p2(\$1)} |
| \$1: CS_CL_EIB__ARG | Option value | cs-ptr-ref |
| CS_CL_ENDACTIVITY | Endactivity Clause (CICS) | ENDACTIVITY |
| CS_CL_EQUAL | Equal Clause (CICS) | EQUAL |
| CS_CL_ERASE | Erase Clause (CICS) | ERASE |
| CS_CL_ERASEAUP | Eraseaup Clause (CICS) | ERASEAUP |
| CS_CL_FCT | Fct Clause (CICS) | FCT |
| CS_CL_FIELD | Field Clause (CICS) | FIELD{\$1?\$p2(\$1)} |
| \$1: CS_CL_FIELD__ARG | Option value | cs-data-area |
| CS_CL_FILE | File Clause (CICS) | FILE{\$1?\$p2(\$1)} |
| \$1: CS_CL_FILE__ARG | Option value | cs-filename |
| CS_CL_FLENGTH | Flength Clause (CICS) | FLENGTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_FLENGTH__ARG | Option value | cs-arg-any |
| CS_CL_FMH | Fmh Clause (CICS) | FMH |
| CS_CL_FOR | For Clause (CICS) | FOR |

| | | |
|-----------------------------|---------------------------|----------------------------|
| CS_CL_FORMFEED | Formfeed Clause (CICS) | FORMFEED |
| CS_CL_FREEKB | Freekb Clause (CICS) | FREEKB |
| CS_CL_FROM | From Clause (CICS) | FROM{\$1?\$p2(\$1)} |
| \$1: CS_CL_FROM__ARG | Option value | cs-data-area |
| CS_CL_FRSET | Frset Clause (CICS) | FRSET |
| CS_CL_FULLDATE | Fulldate Clause (CICS) | FULLDATE{\$1?\$p2(\$1)} |
| \$1: CS_CL_FULLDATE__ARG | Option value | cs-data-area |
| CS_CL_GENERIC | Generic Clause (CICS) | GENERIC |
| CS_CL_GTEQ | Gteq Clause (CICS) | GTEQ |
| CS_CL_HONEOM | Honeom Clause (CICS) | HONEOM |
| CS_CL_HOURS | Hours Clause (CICS) | HOURS{\$1?\$p2(\$1)} |
| \$1: CS_CL_HOURS__ARG | Option value | cs-data-value |
| CS_CL_IMMEDIATE | Immediate Clause (CICS) | IMMEDIATE |
| CS_CL_INITIMG | Initimg Clause (CICS) | INITIMG{\$1?\$p2(\$1)} |
| \$1: CS_CL_INITIMG__ARG | Option value | cs-data-value |
| CS_CL_INPARTN | Inpartn Clause (CICS) | INPARTN{\$1?\$p2(\$1)} |
| \$1: CS_CL_INPARTN__ARG | Option value | cs-name |
| CS_CL_INPUTMSG | Inputmsg Clause (CICS) | INPUTMSG{\$1?\$p2(\$1)} |
| \$1: CS_CL_INPUTMSG__ARG | Option value | cs-data-area |
| CS_CL_INPUTMSGLEN | Inputmsglen Clause (CICS) | INPUTMSGLEN{\$1?\$p2(\$1)} |
| \$1: CS_CL_INPUTMSGLEN__ARG | Option value | cs-data-value |
| CS_CL_INTERVAL | Interval Clause (CICS) | INTERVAL{\$1?\$p2(\$1)} |
| \$1: CS_CL_INTERVAL__ARG | Option value | cs-hhmmss |

| | | |
|----------------------|--------------------|---------------------|
| CS_CL_INT0 | Into Clause (CICS) | INT0{\$1?\$p2(\$1)} |
| \$1: CS_CL_INT0__ARG | Option value | cs-data-area |

| | | |
|--------------|----------------------|--------|
| CS_CL_INVITE | Invite Clause (CICS) | INVITE |
|--------------|----------------------|--------|

| | | |
|----------------------|--------------------|---------------------|
| CS_CL_ITEM | Item Clause (CICS) | ITEM{\$1?\$p2(\$1)} |
| \$1: CS_CL_ITEM__ARG | Option value | cs-arg-any |

| | | |
|---------------------------|-------------------------|--------------------------|
| CS_CL_KEYLENGTH | Keylength Clause (CICS) | KEYLENGTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_KEYLENGTH__ARG | Option value | cs-data-value |

| | | |
|-----------|-------------------|-----|
| CS_CL_L40 | L40 Clause (CICS) | L40 |
|-----------|-------------------|-----|

| | | |
|-----------|-------------------|-----|
| CS_CL_L64 | L64 Clause (CICS) | L64 |
|-----------|-------------------|-----|

| | | |
|-----------|-------------------|-----|
| CS_CL_L80 | L80 Clause (CICS) | L80 |
|-----------|-------------------|-----|

| | | |
|-----------------------|---------------------|----------------------|
| CS_CL_LABEL | Label Clause (CICS) | LABEL{\$1?\$p2(\$1)} |
| \$1: CS_CL_LABEL__ARG | Option value | cs-label |

| | | |
|------------|--------------------|------|
| CS_CL_LAST | Last Clause (CICS) | LAST |
|------------|--------------------|------|

| | | |
|---------------------|-------------------|--------------------|
| CS_CL_LDC | Ldc Clause (CICS) | LDC{\$1?\$p2(\$1)} |
| \$1: CS_CL_LDC__ARG | Option value | cs-name |

| | | |
|---------------|-----------------------|---------|
| CS_CL_LEAVEKB | Leavekb Clause (CICS) | LEAVEKB |
|---------------|-----------------------|---------|

| | | |
|------------------------|----------------------|-----------------------|
| CS_CL_LENGTH | Length Clause (CICS) | LENGTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_LENGTH__ARG | Option value | cs-arg-any |

| | | |
|--------------------------|------------------------|-------------------------|
| CS_CL_LINEADDR | Lineaddr Clause (CICS) | LINEADDR{\$1?\$p2(\$1)} |
| \$1: CS_CL_LINEADDR__ARG | Option value | cs-data-value |

| | | |
|------------|--------------------|------|
| CS_CL_MAIN | Main Clause (CICS) | MAIN |
|------------|--------------------|------|

| | | |
|---------------------|-------------------|--------------------|
| CS_CL_MAP | Map Clause (CICS) | MAP{\$1?\$p2(\$1)} |
| \$1: CS_CL_MAP__ARG | Option value | cs-name |

| | | |
|-----------------------------|---------------------------|----------------------------|
| CS_CL_MAPONLY | Maponly Clause (CICS) | MAPONLY |
| CS_CL_MAPPED | Mapped Clause (CICS) | MAPPED |
| CS_CL_MAPPINGDEV | Mappingdev Clause (CICS) | MAPPINGDEV{\$1?\$p2(\$1)} |
| \$1: CS_CL_MAPPINGDEV__ARG | Option value | cs-data-value |
| CS_CL_MAPSET | Mapset Clause (CICS) | MAPSET{\$1?\$p2(\$1)} |
| \$1: CS_CL_MAPSET__ARG | Option value | cs-name |
| CS_CL_MASSINSERT | Massinsert Clause (CICS) | MASSINSERT |
| CS_CL_MAXLENGTH | Maxlength Clause (CICS) | MAXLENGTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_MAXLENGTH__ARG | Option value | cs-data-value |
| CS_CL_MAXLENGTH | Maxlength Clause (CICS) | MAXLENGTH{\$1?\$p2(\$1)} |
| \$1: CS_CL_MAXLENGTH__ARG | Option value | cs-arg-any |
| CS_CL_MINUTES | Minutes Clause (CICS) | MINUTES{\$1?\$p2(\$1)} |
| \$1: CS_CL_MINUTES__ARG | Option value | cs-data-value |
| CS_CL_MMDDYY | Mmddyy Clause (CICS) | MMDDYY{\$1?\$p2(\$1)} |
| \$1: CS_CL_MMDDYY__ARG | Option value | cs-data-area |
| CS_CL_MMDDYYYY | Mmddyyyy Clause (CICS) | MMDDYYYY{\$1?\$p2(\$1)} |
| \$1: CS_CL_MMDDYYYY__ARG | Option value | cs-data-area |
| CS_CL_MONTHOFYEAR | Monthofyear Clause (CICS) | MONTHOFYEAR{\$1?\$p2(\$1)} |
| \$1: CS_CL_MONTHOFYEAR__ARG | Option value | cs-data-area |
| CS_CL_NEXT | Next Clause (CICS) | NEXT |
| CS_CL_NOCHECK | Nocheck Clause (CICS) | NOCHECK |
| CS_CL_NODUMP | Nodump Clause (CICS) | NODUMP |
| CS_CL_NOEDIT | Noedit Clause (CICS) | NOEDIT |

| | | |
|--------------------------|---------------------------|-------------------------|
| CS_CL_NOHANDLE | Nohandle Clause (CICS) | NOHANDLE |
| CS_CL_NOSUSPEND | Nosuspend Clause (CICS) | NOSUSPEND |
| CS_CL_NOTRUNCATE | Nottruncate Clause (CICS) | NOTRUNCATE |
| CS_CL_NUMITEMS | Numitems Clause (CICS) | NUMITEMS{\$1?\$p2(\$1)} |
| \$1: CS_CL_NUMITEMS__ARG | Option value | cs-data-area |
| CS_CL_NUMREC | Numrec Clause (CICS) | NUMREC{\$1?\$p2(\$1)} |
| \$1: CS_CL_NUMREC__ARG | Option value | cs-arg-any |
| CS_CL_OUTPARTN | Outpartn Clause (CICS) | OUTPARTN{\$1?\$p2(\$1)} |
| \$1: CS_CL_OUTPARTN__ARG | Option value | cs-name |
| CS_CL_PAGING | Paging Clause (CICS) | PAGING |
| CS_CL_PASSBK | Passbk Clause (CICS) | PASSBK |
| CS_CL_PCT | Pct Clause (CICS) | PCT |
| CS_CL_PPT | Ppt Clause (CICS) | PPT |
| CS_CL_PRINT | Print Clause (CICS) | PRINT |
| CS_CL_PROGRAM | Program Clause (CICS) | PROGRAM{\$1?\$p2(\$1)} |
| \$1: CS_CL_PROGRAM__ARG | Option value | cs-name NULL |
| CS_CL_PROTECT | Protect Clause (CICS) | PROTECT |
| CS_CL_PSEUDOBIN | Pseudobin Clause (CICS) | PSEUDOBIN |
| CS_CL_QNAME | Qname Clause (CICS) | QNAME{\$1?\$p2(\$1)} |
| \$1: CS_CL_QNAME__ARG | Option value | cs-name |

| | | |
|-----------------------|---------------------|----------------------|
| CS_CL_QUEUE | Queue Clause (CICS) | QUEUE{\$1?\$p2(\$1)} |
| \$1: CS_CL_QUEUE__ARG | Option value | cs-arg-any |

| | | |
|-----------|-------------------|-----|
| CS_CL_RBA | Rba Clause (CICS) | RBA |
|-----------|-------------------|-----|

| | | |
|------------------|--------------------------|------------|
| CS_CL_REPEATABLE | Repeatable Clause (CICS) | REPEATABLE |
|------------------|--------------------------|------------|

| | | |
|-----------------------|---------------------|----------------------|
| CS_CL_REQID | Reqid Clause (CICS) | REQID{\$1?\$p2(\$1)} |
| \$1: CS_CL_REQID__ARG | Option value | cs-arg-any |

| | | |
|-------------|---------------------|-------|
| CS_CL_RESET | Reset Clause (CICS) | RESET |
|-------------|---------------------|-------|

| | | |
|----------------------|--------------------|---------------------|
| CS_CL_RESP | Resp Clause (CICS) | RESP{\$1?\$p2(\$1)} |
| \$1: CS_CL_RESP__ARG | Option value | cs-data-area |

| | | |
|-----------------------|---------------------|----------------------|
| CS_CL_RESP2 | Resp2 Clause (CICS) | RESP2{\$1?\$p2(\$1)} |
| \$1: CS_CL_RESP2__ARG | Option value | cs-data-area |

| | | |
|---------------|-----------------------|---------|
| CS_CL_REWRITE | Rewrite Clause (CICS) | REWRITE |
|---------------|-----------------------|---------|

| | | |
|------------------------|----------------------|-----------------------|
| CS_CL_RIDFLD | Ridfld Clause (CICS) | RIDFLD{\$1?\$p2(\$1)} |
| \$1: CS_CL_RIDFLD__ARG | Option value | cs-data-area |

| | | |
|-----------|-------------------|-----|
| CS_CL_RRN | Rrn Clause (CICS) | RRN |
|-----------|-------------------|-----|

| | | |
|-------------------------|-----------------------|------------------------|
| CS_CL_RTERMID | Rtermid Clause (CICS) | RTERMID{\$1?\$p2(\$1)} |
| \$1: CS_CL_RTERMID__ARG | Option value | cs-arg-any |

| | | |
|--------------------------|------------------------|-------------------------|
| CS_CL_RTRANSID | Rtransid Clause (CICS) | RTRANSID{\$1?\$p2(\$1)} |
| \$1: CS_CL_RTRANSID__ARG | Option value | cs-arg-any |

| | | |
|-------------------------|-----------------------|------------------------|
| CS_CL_SECONDS | Seconds Clause (CICS) | SECONDS{\$1?\$p2(\$1)} |
| \$1: CS_CL_SECONDS__ARG | Option value | cs-data-value |

| | | |
|-------------------------|-----------------------|------------------------|
| CS_CL_SESSION | Session Clause (CICS) | SESSION{\$1?\$p2(\$1)} |
| \$1: CS_CL_SESSION__ARG | Option value | cs-name |

| | | |
|---------------------|-------------------|--------------------|
| CS_CL_SET | Set Clause (CICS) | SET{\$1?\$p2(\$1)} |
| \$1: CS_CL_SET__ARG | Option value | cs-arg-any |

| | | |
|-------------------------|----------------------------|------------------------|
| CS_CL_SHARED | Shared Clause (CICS) | SHARED |
| CS_CL_SIT | Sit Clause (CICS) | SIT |
| CS_CL_STATE | State Clause (CICS) | STATE{\$1?\$p2(\$1)} |
| \$1: CS_CL_STATE__ARG | Option value | cs-arg-any |
| CS_CL_STORAGE | Storage Clause (CICS) | STORAGE |
| CS_CL_STRFIELD | Strfield Clause (CICS) | STRFIELD |
| CS_CL_SYNCONRETURN | Synconreturn Clause (CICS) | SYNCONRETURN |
| CS_CL_SYSID | Sysid Clause (CICS) | SYSID{\$1?\$p2(\$1)} |
| \$1: CS_CL_SYSID__ARG | Option value | cs-systemname |
| CS_CL_TABLES | Tables Clause (CICS) | TABLES |
| CS_CL_TASK | Task Clause (CICS) | TASK |
| CS_CL_TCT | Tct Clause (CICS) | TCT |
| CS_CL_TCTUA | Tctua Clause (CICS) | TCTUA{\$1?\$p2(\$1)} |
| \$1: CS_CL_TCTUA__ARG | Option value | cs-ptr-ref |
| CS_CL_TERMID | Termid Clause (CICS) | TERMID{\$1?\$p2(\$1)} |
| \$1: CS_CL_TERMID__ARG | Option value | cs-arg-any |
| CS_CL_TERMINAL | Terminal Clause (CICS) | TERMINAL |
| CS_CL_TIME | Time Clause (CICS) | TIME{\$1?\$p2(\$1)} |
| \$1: CS_CL_TIME__ARG | Option value | cs-arg-any |
| CS_CL_TIMESEP | Timesep Clause (CICS) | TIMESEP{\$1?\$p2(\$1)} |
| \$1: CS_CL_TIMESEP__ARG | Option value | cs-data-value NULL |

| | | |
|-------------------------|---------------------------|------------------------|
| CS_CL_TOKEN | Token Clause (CICS) | TOKEN{\$1?\$p2(\$1)} |
| \$1: CS_CL_TOKEN__ARG | Option value | cs-arg-any |
| CS_CL_TRANSID | Transid Clause (CICS) | TRANSID{\$1?\$p2(\$1)} |
| \$1: CS_CL_TRANSID__ARG | Option value | cs-name |
| CS_CL_TWA | Twa Clause (CICS) | TWA{\$1?\$p2(\$1)} |
| \$1: CS_CL_TWA__ARG | Option value | cs-ptr-ref |
| CS_CL_UNCOMMITTED | Uncommitted Clause (CICS) | UNCOMMITTED |
| CS_CL_UNTIL | Until Clause (CICS) | UNTIL |
| CS_CL_UPDATE | Update Clause (CICS) | UPDATE{\$1?\$p2(\$1)} |
| \$1: CS_CL_UPDATE__ARG | Option value | cs-cvda NULL |
| CS_CL_USERDATAKEY | Userdatakey Clause (CICS) | USERDATAKEY |
| CS_CL_USERID | Userid Clause (CICS) | USERID{\$1?\$p2(\$1)} |
| \$1: CS_CL_USERID__ARG | Option value | cs-arg-any |
| CS_CL_USING | Using Clause (CICS) | USING{\$1?\$p2(\$1)} |
| \$1: CS_CL_USING__ARG | Option value | cs-arg-any |
| CS_CL_WAIT | Wait Clause (CICS) | WAIT |
| CS_CL_YEAR | Year Clause (CICS) | YEAR{\$1?\$p2(\$1)} |
| \$1: CS_CL_YEAR__ARG | Option value | cs-data-area |
| CS_CL_YYDDD | Yydd Clause (CICS) | YYDDD{\$1?\$p2(\$1)} |
| \$1: CS_CL_YYDDD__ARG | Option value | cs-data-area |
| CS_CL_YYDDMM | Yyddmm Clause (CICS) | YYDDMM{\$1?\$p2(\$1)} |
| \$1: CS_CL_YYDDMM__ARG | Option value | cs-data-area |
| CS_CL_YYMDD | Yymmdd Clause (CICS) | YYMDD{\$1?\$p2(\$1)} |
| \$1: CS_CL_YYMDD__ARG | Option value | cs-data-area |
| CS_CL_YYYYDDD | Yyyddd Clause (CICS) | YYYYDDD{\$1?\$p2(\$1)} |
| \$1: CS_CL_YYYYDDD__ARG | Option value | cs-data-area |

| | | |
|--------------------------|-----------------------|-------------------------|
| CS_CL_YYYYDDMM | Yyyddmm Clause (CICS) | YYYYDDMM{\$1?\$p2(\$1)} |
| \$1: CS_CL_YYYYDDMM__ARG | Option value | cs-data-area |

| | | |
|--------------------------|------------------------|-------------------------|
| CS_CL_YYYYMMDD | Yyyymmdd Clause (CICS) | YYYYMMDD{\$1?\$p2(\$1)} |
| \$1: CS_CL_YYYYMMDD__ARG | Option value | cs-data-area |

10.10.3 Arguments

| | | |
|--------------|-----------------------------------|--|
| • cs-arg-any | Allowed arguments of CICS clauses | SR_ADDRESS_OF (identifier) SR_LENGTH_OF (identifier) ident-liter |
|--------------|-----------------------------------|--|

| | | |
|-----------------|-----------------------------------|-------------|
| • cs-data-value | Allowed arguments of CICS clauses | ident-liter |
|-----------------|-----------------------------------|-------------|

| | | |
|----------------|-----------------------------------|------------|
| • cs-data-area | Allowed arguments of CICS clauses | identifier |
|----------------|-----------------------------------|------------|

| | | |
|----------------|-----------------------------------|-------------|
| • cs-ptr-value | Allowed arguments of CICS clauses | ident-liter |
|----------------|-----------------------------------|-------------|

| | | |
|--------------|-----------------------------------|---|
| • cs-ptr-ref | Allowed arguments of CICS clauses | SR_ADDRESS_OF (identifier) identifier |
|--------------|-----------------------------------|---|

| | | |
|-----------|-----------------------------------|-------------|
| • cs-cvda | Allowed arguments of CICS clauses | ident-liter |
|-----------|-----------------------------------|-------------|

| | | |
|-----------|-----------------------------------|-------------|
| • cs-name | Allowed arguments of CICS clauses | ident-liter |
|-----------|-----------------------------------|-------------|

| | | |
|-------------|-----------------------------------|-------------|
| • cs-hhmmss | Allowed arguments of CICS clauses | ident-liter |
|-------------|-----------------------------------|-------------|

| | | |
|---------------|-----------------------------------|-------------|
| • cs-filename | Allowed arguments of CICS clauses | ident-liter |
|---------------|-----------------------------------|-------------|

| | | |
|-----------------|-----------------------------------|-------------|
| • cs-systemname | Allowed arguments of CICS clauses | ident-liter |
|-----------------|-----------------------------------|-------------|

| | | |
|------------|-----------------------------------|---------------|
| • cs-label | Allowed arguments of CICS clauses | NAMED_OBJ_USE |
|------------|-----------------------------------|---------------|

| | | |
|----------------------------|------------------------------------|--|
| • destid-spec | Destination ID specification | NULL |
| • options-spec | Options specification | NULL |
| • subaddr-spec | Address specification | NULL |
| • optional-data-value-spec | Optional data-value parameter | NULL |
| • assignment-list | List of assignments in ASSIGN stmt | CS_ASSIGNMENT_LIST (CS_GRP_ASSIGN) |
| • condition-handler-list | List of CICS condition handlers | CS_COND_HANDLER_LIST (CS_GRP_CONDITION_HANDLER) |
| • aid-handler-list | List of AID handlers | CS_COND_HANDLER_LIST (CS_GRP_CONDITION_HANDLER) |
| CS_ASSIGNMENT_LIST | Assignment List | @\n |
| CS_GRP_ASSIGN | Get Env value (CICS) | \$1{\$2=\$p2(\$2)} |
| \$1: ASS_ENV_NAME | Env var name | CHAR_STRING |
| \$2: ASS_VAR_NAME | Variable to store env value | ident-liter |
| CS_COND_HANDLER_LIST | Condition Handler List | @\n |
| CS_GRP_CONDITION_HANDLER | Condition Handler (CICS) | \$1{\$2?\$p2(\$2)} |
| \$1: HND_CONDNAME | Condition name | CHAR_STRING |
| \$2: HND_PROCNAME | Procedure name | cs-label |
| • SECTION_SCHEMA | externally defined | . |
| • EX_DML_ALL | externally defined | . |
| • EX_DML_SELECTIVE | externally defined | . |
| • EX_DML_ONLY | externally defined | . |
| LOG_NOT | Not | NOT \$1 |
| \$1: LOG_NOT__LOGNOT_ARG1 | Arg1 | condition |

Bibliography

- [Ame85] American National Standards Institute. *ANSI-85 Cobol Standard*, 1985.
- [ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison Wesley, 1986.
- [Int] International Business Machines Corp. *VS COBOL II Language Reference Manual*.
- [Mic93] MicroFocus. *MicroFocus Cobol Language Reference*, 1993.