**TypeCobol Compiler – Initial requirements**

# Project Goals

## COBOL Front-end

### Compatibility

G1. Full compatibility with IBM Enterprise Cobol 5.1 for zOS

G2. Full compatibility with DB2 11 for zOs - Cobol compiler coprocessor   
 Approximate support for DB2 11 for the Cobol precompiler

G3. Do not support obsolete features when they would have a huge impact on performance

G4. Do not prepare for the simultaneous support of multiple Cobol versions in the compiler

### Text sources

G3. EBCDIC source  
- reference format only   
- fixed-length lines (no line delimiter chars)  
- no support for DBCS chars in source text  
 \* accept syntax for DBCS literals   
 \* but raise an error whenever a shift-in or shift-out character is detected

G4.ASCII or Unicode source  
- reference format or free format (see Extensions)  
- line delimiter chars  
- no limitation on the charset used for user defined words or literals

G5. Model the text sources as a set update events, consecutive lines input in the case of batch compilation, but also random updates in the case of an editor

G6. Be resilient to incomplete or bad syntax, to enable on-the-fly validation during development

G7. Do not handle compilation units in isolation, but introduce a project system which enables the compiler to access to all secondary compilation units, to share the relevant information (data structure and entry points), and to implement cross-compilation units checks

### Performances

G8. Incremental lexer, parser and type checker to enable real-time code analysis in an editor

G9. Ensure good locality of the validations to minimize the number of checks for each incremental update

G10. Provide options to disable the last phases of validation and/or the most costly checks for a fast but approximate analysis

### API

G11. Expose all information known at each step of the compiler pipeline with a friendly public API, not just the end result, so that all tools can find the relevant information they need

G12. Provide a subset of the compiler specialized for COPYbook analysis

G13. Provide a clean and complete documentation of the grammar, the code model used, and the API

G14. Display rich error messages, always with a quote to the IBM reference documentation

G15. Provide a complete source code listing feature, with several formats  
- page-based standard text format described in the specification  
- HTML or XAML pretty-printed source code before or after the precompiler step  
- graphic syntax explorer side-by-side with the source code

G16. Ship a complete unit tests suite and tests results with each compiler version

G17. Enable integration :  
- of the project system with DB2 database schemas or DDL files  
- with a SQL compiler frontend to analyze embedded EXEC SQL statement

G18. Execution flow and data flow analysis : provide code representation enabling impact analysis and refactoring   
- support indirect calls analysis  
- provide a pluggable mechanism for custom indirect code resolution  
- provide extension points to enable integration with external directories (data or services)

G19. Syntax pattern matching : provide a framework enabling easy development of coding standards checks

### Extensions

G20. For each extension, always provide a service with takes the extended source text as an input and generates a standards compliant Cobol program as the output

G21. As an option, support entreprise-specific precompiler steps  
- handle Euro Information - specific REMARKS/COPY statements

G22. Support free format source text  
- unlimited line length  
- no reliance on columns numbers  
- introduce special tokens to replace column numbers when necessary   
 (ex : debug or continuation lines)

G23. As an option, support identifier extensions  
- no limit on the length of the data names  
- names qualification to enable reuse of complete data structures  
- enforce limits only on the identifiers shared with other programs

G24. As an option, support richer type annotations and use them to check the validity of MOVE statements and program CALLs. Expose this type information to client tools.

G25. As an option, implement strict type checking for all programs and procedure calls

## COBOL dev Tools

### Batch compiler

G24. Provide a console based batch compiler with limited diagnostics capabilities (only error messages)

### Syntax explorer

G25. Provide an exhaustive visual tool which enables syntax discovery and compiler diagnostics :  
- tokens  
- abstract syntax tree  
- code model  
- list of all validation rules

### Editor

G26. Provide a fast and light Cobol code editor, with no dependencies on Eclipse or Visual Studio, but wich could be embedded in either one of them if necessary  
- syntax highlighting  
- on-the-fly error reporting   
- code completion  
- file outline and code folding  
- navigate references  
- automatic formatting  
- auto fixes / refactoring

### IDE

G27. Provide an IDE which enables navigation and comparisons between all the files mentioned in the project system

G28. As an option, provide integration between the IDE and Euro Information tools