Appendix A: Grammar of Configuration Files

In Section 6.2, we described how configuration (text) files are used to configure object manager components in an enterprise application's process topology. This appendix gives a grammar for such files. We present the grammar as we defined it for the ANTLR parser/scanner generator used for our implementation of the *Free Data Objects* framework. For better readability, only production rules for the parser are given – details for the scanner, ANTLR directives, and all code generation related parts are omitted.

In production rules, the characters *, ?, |, (,), and " have special meaning:

`(` *expr* `)*` produces *expr* 0..n times.
`(` *expr* `)?` produces *expr* 0..1 times.
expr1 `|` *expr2* produces either *expr1* or *expr2*.
"*lit*" produces the literal *lit*.

The rule `startRule` describes the structure of an object manager configuration file. The rule `query_def` describes the structure of a query string. The rule `bool_or_expr` describes the structure of an arbitrarily nested Boolean expression string. Query strings and Boolean expression strings may occur in object manager configuration files. However, they may also be parsed as separate entities during normal transaction processing when queries are executed and predicate checks are applied to data objects, respectively.

```
startRule: ( object_manager_def )*

object_manager_def: "OBJECT" "MANAGER" object_manager_name
         LCURLY
         ( object_manager_id )?
         ( "EXPORTS" EQUALS import_export_expr SEMI )?
         ( data_object_def | interface_def | domain_def
           | server_connection_def | client_listener_def )*
         RCURLY

object_manager_name: ASTERISK | IDENT

object_manager_id: "ID" EQUALS NUMBER SEMI

import_export_expr: ( import_export_component )*

import_export_component: LPAREN IDENT ( COMMA )? IDENT RPAREN

data_object_def: "DATA" "OBJECT" IDENT
         LCURLY
         "TYPECODE" EQUALS NUMBER SEMI
         ( attribute_def )*
         RCURLY

attribute_def: IDENT ( cluster_flag )? ( index_hint_flag )? COLON IDENT SEMI

cluster_flag: EXCLAMATION_MARK

index_hint_flag: ASTERISK

interface_def: "INTERFACE" "FOR" IDENT COLON qualified_class_name SEMI

qualified_class_name: qualified_class_component ( DOT qualified_class_component )*

qualified_class_component: IDENT
```

```
domain_def: "DOMAIN" IDENT
         LCURLY
         "CODE" EQUALS NUMBER SEMI
         "DEF" EQUALS query_def ( COMMA query_def )* SEMI
         "RITREE" EQUALS ritree_expr SEMI
         RCURLY

query_def: "SELECT" "FROM" IDENT ( where_bool_expr )? ( order_by_def )?

order_by_def: "ORDER BY" IDENT ( sort_order_def )?

sort_order_def: "ASC" | "DESC"

where_bool_expr: "WHERE" bool_or_expr

bool_or_expr: bool_and_expr ( "OR" bool_and_expr )*

bool_and_expr: bool_simple_expr ( "AND" bool_simple_expr )*

bool_simple_expr:
           "TRUE"
         | "FALSE"
         | "NOT" bool_or_expr
         | IDENT IS_NULL
         | IDENT IS_NOT_NULL
         | LPAREN bool_or_expr RPAREN
         | IDENT compare_op const_value
         | const_value compare_op IDENT

const_value:
           NUMBER
         | SINGLE_QUOTE_STRING_LITERAL
         | "TRUE"
         | "FALSE"
         | "NULL"

compare_op: EQUALS | NOT_EQUALS | GE | GT | LE | LT

ritree_expr:
           R_LPAREN ritree_expr ( COMMA ritree_expr )* RPAREN
         | I_LPAREN ritree_expr ( COMMA ritree_expr )* RPAREN
         | IDENT

server_connection_def: "SERVER" "CONNECTION"
         LCURLY
         "SERVERADDRESS" EQUALS address SEMI
         "IMPORTS" EQUALS import_export_expr SEMI
         "ADAPTER" EQUALS IDENT SEMI
         RCURLY

address: IDENT ( address_param )*

address_param: DOUBLE_QUOTE_STRING_LITERAL

client_listener_def: "CLIENT" "LISTENER"
         LCURLY
         "LOCALADDRESS" EQUALS address SEMI
         "ADAPTER" EQUALS IDENT SEMI
         RCURLY
```

The grammar above uses the following tokens defined by the scanner:

```
ASTERISK, COLON, COMMA, DOT, DOUBLE_QUOTE_STRING_LITERAL, EQUALS, EXCLAMATION_MARK, GE,
GT, I_LPAREN, IDENT, IS_NOT_NULL, IS_NULL, LCURLY, LE, LPAREN, LT, NOT_EQUALS, NUMBER,
ORDER_BY, R_LPAREN, RCURLY, RPAREN, SEMI, SINGLE_QUOTE_STRING_LITERAL
```