Virtual DDBMS C++ Direct-Engine-Call API Documentation Draft v0.8.0.0

The public API consist of three classes:

- **Exception**
- **DBRow**
- **DDBMSConnection**

The Exception Class

Below are the public interface of the class :

```
class Exception : public exception {
    public:
        // Construct a new exception object using the given informations
        Exception(const string& what, const string& func, const string& file, int32_t line);

        // Standard dtor
        virtual ~Exception() throw();

        // Getters
        const char*   what() const throw();
        const char*   func() const throw();
        const char*   file() const throw();
              int32_t line() const throw();

        // Returns the full exception text
        const string fullText(size_t indent = 0) const;

        // Returns the full exception text from the given std::exception object
        static const string fullText(const exception& e, size_t indent = 0);

        // Print the exception information
        void print() const;

        // Print the exception information from the given std::exception object
        static void print(const exception& e);
    };
```

The class interface should be self-explanatory.

The DBRow Class

Below are the public interface of the class :

```
class DBRow {
    public:
        // Standard dtor
        virtual ~DBRow() throw();

        // Returns 'true' if the row is empty (contains no field)
        size_t empty() const;

        // Returns the number of fields in the row
        size_t size() const;

        // Returns the whole vector of NULLs' marker
        const vector<bool>& vectNULL() const;

        // Returns the whole vector of fields' data
        const vector<string>& vectField() const;

        // Returns 'true' if the indicated field contains a NULL
        bool isNULL(size_t index) const;

        // Returns the content of the indicated field
        const string operator[](size_t index) const;

        // Swap the contents of the row
        void swap(DBRow& dst);
};
```

The class interface should be self-explanatory.

The DBMSConnection Class

Below are the public interface of the class :

```
class DDBMSConnection {
    public:
        // Construct a new DDBMS connection object
        DDBMSConnection(const string& userID,
                        const string& primaryCatalogType,
                              int     primaryCatalogPort,
                              bool    canUpdateRecordsInTables         = false,
                              bool    canInsertDeleteRecordsFromTables = false,
                              bool    canShowTables                    = false,
                              bool    canCreateDeleteTables            = false,
                              bool    canShowDatabases                 = false,
                              bool    canCreateDeleteDatabases         = false,
                              bool    canGrantAccess                   = false,
                              bool    canShowResources                 = false,
                              bool    canAddRemoveResources            = false,
                              bool    canShowExecSProcFunc             = false,
                              bool    canAddRemoveSProcFunc            = false,
                              int     lockManagerServicePort           = 0);

        // Standard dtor
        ~DDBMSConnection();

        // Get the engine version
        void ddbmsEngineVersion(uint8_t& major, uint8_t& minor, uint8_t& revision,
                                uint8_t& bugFix) const;

        // Evaluate the given SQL statements; use the function fetchNextRow() to get the
        // the evaluation result
        void evalSQL(const string& sqlStatement);

        // Execute the given SQL statements; in case the statements produce row sets, use
        // the function fetchNextRow() to collect them
        void execSQL(const string& sqlStatements);

        // Prepare the given SQL statement
        void prepSQL(const string& sqlStatement);

        // Set values of the prepared SQL
        void prepSetVal(char id, const string& value);

        // Execute the prepared SQL
        void prepExec();

        // Fetch the next row; returns 'false' if no more row can be fetched
        bool fetchNextRow(DBRow& row);

        // Discard all pending rows
        void discardAll();

        // Delete all data in the primary catalog (it most likely not something you really
        // want to do!)
        void purgeAllResourceCatalogData();

        // Call this to ask the DDBMS engine to reload the resource catalog data from the
        // original database
        void reloadResourceCatalog(bool readLock = true);

        // Add a new resource definition
        const string addResource(const string& type,        const string& host,
                                       int     port,        const string& dbName,
                                 const string& user,        const string& password,
                                 const string& odbcOptions, const string& description);

        // Delete a resource definition (it maybe not something you really want to do!)
        void removeResource(const string& resID);
```

```
        // Modify a resource definition (it maybe not something you really want to do!)
        void modResource(const string& resID,
                         const string& type,        const string& host,
                                       int     port,        const string& dbName,
                                 const string& user,        const string& password,
                                 const string& odbcOptions, const string& description);

        // Direct read only access to the resource catalog
        const ResourceCatalog& resourceCatalog() const;
};
```

The class interface should be self-explanatory.