MaXXsettings

	Top	

Introduction

MaXX**settings** is a dynamic configuration management subsystem designed from the ground up with simplicity in mind while not sacrificing flexibility and extensibility. MaXX**settings** comes with its own CLI interface allowing simple management, automation via scripting, inline-query and easy application integration. MaXX**settings** also provides Java and C++ binding making it super easy to integrate within most modern applications. MaXX **settings** allow the definition of System wide setting, we call them **Instruments**, and user's overridables called **User Preferences**.

Here are the requirements that MaXXsettings must strive to enforce or provide:

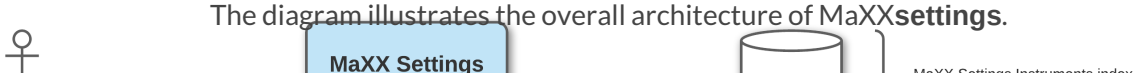
- Retrieve information as fast as possible (flat lookup speed curve).
- Provide different levels of verbosity (admin vs normal user).
- Software design based on current/modern technologies while future proofing the code with a component/modular approach.
- Use SOLID Principles (most): Single responsibility, open-close, interface segregation and dependency inversion.
- Favour simplicity over complexity.
- Support multiple OS.
- Provide a Command Line Interface (CLI) to administer, query and set data.
- Be human friendly with its interfaces.
- Provide an API for C++ and Java clients.
- Provide user based authentication.
- Support UTF-16 for its internal String encoding.
- Support hierarchical data structure suited for a dynamic typed configuration management system for Desktop, Application and FileType instrumentation.

For the complete Technical Specifications, refer to this [online] document

Design

The Architecture design on which MaXX**settings** is built follows the Client/Server model, where the Client is represented by Users using a CLI type interface or Applications using an API both interacting with the Server. The Server provides the functionality to manage instrumentation on behalf of the Client.

The diagram illustrates the overall architecture of MaXX**settings**.



Development and Execution Platform

Java was selected for the first implementation of the Server for its richness in features, robustness, maturity and special affinity with server/service APIs and prebuilt components. The GraalVM was selected for its ability to compile Java byte-code into native code and run applications much faster. The optimizations offered by GraalVM have the added benefits to reduce startup and execution speed quite considerably.

Instrumentation Data Persistence

All information managed by the Server is persisted into regular files. No external dependency required for the database.

Database

The database is implemented using fixed length field strategy to ensure simplicity but high performance. Saved information into the database is kept to a minimum as it is mostly to provide lookup mechanism.

Development

MaXX**settings** can be integrated using the CLI command line or via its C/C++/Java APIs.

Refer to MaXXsettings Framework documentation for more detail.

Top

Revision #15
Created 16 January 2021 16:20:25 by Eric Masson
Updated 24 June 2021 12:27:02 by Eric Masson