# OSP XMLtoSQL Extended Edition
# Whats New?

Theodore Olsauskas-Warren – U5195918
October 2014

**Introduction**

This document serves to outline the changes that the XMLtoSQL component of OSP underwent in it's transition from the Minimum Viable Product (MVP) to the Extended Edition (EE). It will discuss changes in two main areas, the database schema it uses and error handling and detection. This report assumes the reader has read the technical documentation for the MVP, as well as the schema exploration documentation.

**Extended Schema**

The primary difference between the EE and the MVP is the adoption of the extended schema, as outlines in the schema exploration document. This schema allows more relevant information to be stored, thus providing a fuller perspective of the test results to the user. It also streamlines extraction of important information as certain properties, such as collection pass percentages, to be collated at insertion time, rather than calculated later.

The EE has used the same general structure for insertion into the database as the MVP. Information is extracted from the XML files and stored in Python objects that represent that data in a format similar to what will be inserted into the database. Classes represent different tables within the database. This design was present in the definition class in the MVP (only here as definitions was the only table) while in the EE it has extended to cover collections, groups, machines, tests and definitions. Each of these is a class within Python that contains a copy of it's own schema. The SQL interface treats all classes the same way, thus preserving the core design idea that adding attributes to tables is as simple as altering the schema present within the classes themselves, then updating their creation methods; no alteration of the SQL interface is required.

**Error Handling and Logging**

The MVP contained almost no concept of error handling, exception were simply thrown and the program exited. The EE has a drastically increased focus on catching, identifying and reporting on any errors that it encounters. This can be seen in the significant use of the logging module throughout the program, almost no step is undertaken without some form of logging surrounding it. Normally this would cause an overabundance of information, but this is carefully controller by verbosity levels. The user is able to increase or decrease the verbosity level depending on their debugging needs, with the highest level representing a full debugging report where no action is left un-logged.

The EE takes an approach to error handling of maximum reporting to the user, whilst performing minimal attempts to fix any issues. This is by design, as the input we are expecting to be of high quality (it should have come directly from the OpenSCAP program) any errors are considered anomalous. On the other side, by using a tried and tested database and database interface, no attempt is made to correct integrity errors in the database. OSP will create the tables it requires in the database it is pointed to, if however those tables have differing schemas, OSP will completely fail and not attempt to alter or insert other data.

OSP follows the philosophy of failing early, but it creates the idea of file atomicity. That is, significant errors in parsing, extracting or inserting a file cause the entire process to skip that file. Thus, a file is either completely successful, or no portion of it is inserted into the database at all. This means that the user need not be concerned with partial data insertion, they can be sure an entire file has been committed to the database.

**Testing Methodology**

The testing methodology applied to the MVP was somewhat haphazard, tests were exclusively performed manually. This has changed with the EE, which comes with an external testing framework designed to examine it's correctness. The primary motivation behind the tests is to make OSP fail in a specific way, then ensure that OSP has correctly and informatively reported that failure. This methodology reflects the philosophy above of failing, rather than attempting to repair the input. More information about the testing framework can be found in the associated documentation.

**Conclusion**

Overall the EE represents an incremental, evolutionary step for OSP. When compared to the MVP it is a significantly more robust product, the addition of significant logging and testing, has given us significant confidence in its reliability. The move to the extended schema also helps set the EE apart as a more developed, mature product. Whilst there are no significant changes to it's functionality; the core task it performs is nigh identical, it is definitely a significantly improved product when compared to it's predecessor.