SQL OVAL Extension Proposal
September 25, 2015

Background

Due to the need for automated compliance and vulnerability checks against SQL database management systems and a lack of support for the proposed *sql511* test in OVAL 5.11, SPAWAR submits this new proposal for an OVAL extension which addresses the *sql511* deficiencies. The current working designation for this proposal is "*sqlext*" (**SQL** Test for the OVAL **Ext**ension Schema).This is subject to change based upon discussion with the OVAL community regarding standard naming conventions for OVAL extensions.

New Capabilities that Cannot Currently be Accomplished with OVAL

The proposed *sqlext_test* improves upon the *sql57_test*'s data store targeting capabilities and system characteristics granularity. Often, content authors can predict neither DBMS instance names nor names of the databases hosted within those instances. Therefore, it is desirable to write SQL content that expresses generic targeting specifications where the traditional connection string is handled transparently by the scanner tool, itself. The *sqlext_test* enables content authors to provide streamlined, generalized content that can target any DBMS instances and named databases on a given host system without exposing sensitive DBMS configuration details.

In addition to the challenge of characterizing DBMS installations for evaluation, reporting SQL DBMS evaluation results is often challenging because enterprise-grade DBMSs (e.g. Microsoft SQL Server, Oracle Database) support hosting many potential scan targets on a single system. Since modern DBMSs are configured at the host, instance and database levels, it is desirable to access detailed reports for each level, individually. Previous OVAL SQL implementations conflate system characteristics collected from all targeted instances and databases into a single result set, leading to evaluations being resolved into a single vague "pass/fail" result. For example, given a *sql57* check executed against a DBMS server host where a dozen instances are compliant and one is non-compliant, the overall result would be a failure even though there is only a single non-compliant instance. The proposed OVAL *sqlext_test* improves reporting clarity by introducing a new tool behavior that supports a one-to-many relationship between a target system and DBMS instance, and database result sets. In other words, *sql_ext* supports generating multiple OVAL outputs per target system where each output is dedicated to a single DBMS instance or named SQL database.

Relevance of the New Capability

Previous proposals for a SQL OVAL test have been found deficient by the OVAL community. This proposal addresses those deficiencies by abstracting DBMS instance and database identification, and enables the targeting at the instance or database level, and facilitates the production of target-level results sets.

Impact upon OVAL Content Developers

The proposed *sqlext_test* diverges from the *sql* and *sql57* OVAL tests by deprecating the *connection_string* object, state, and item entities in favor of specialized *instance* and *database* string entities. These directives provide content authors with access to all of the usual string-type object entity operations such as *equals, not equal,* and *pattern match*. For example, wildcard pattern matching using the well-known ".*" regular expression is supported for collecting system characteristics from any available DBMS instance or named SQL database that is being hosted by the specified SQL engine. Additionally, the *database* and *sql* object entities have been enhanced with

support for the *xsi:nil* property. A nil *database* entity targets the default database of the specified instance(s) (which is usually the “master” database in MS SQL Server). A nil *sql* entity is a directive to forgo the usual collection of SQL query results and instead determine the existence of the specified instance(s) and database(s). Finally, the *sqlext_object* has been augmented with a *behaviors* entity that declares a *context* attribute. The *context* behavior enumeration type supports three declarations: host (the default value), instance, and database. The default *host* context reflects *sql* and *sql57* probe functionality by aggregating all relevant SQL system characteristics that are available on the given target system into a single consolidated result set. The *instance* and *database* context directives, however, instruct the scanning tool to consider each DBMS instance or named database respectively as its own target system rather than the entire host machine, producing a more granular result set for each targeted data store.

Impact upon OVAL Content Consumers

The *sqlext* OVAL test addresses a known *sql* and *sql57* issue regarding the granularity and clarity of collected system characteristics and compliance results. OVAL content consumers will benefit from the adoption of *sqlext* content by having access to concise, meaningful output that will enable IA specialists and DB administrators to readily isolate and remediate non-compliant DBMS configurations. Furthermore, the proposed *sqlext* OVAL test has been engineered specifically for the purpose of maintaining compatibility with consumers’ current output analysis systems. All XML outputs are schema-valid and consistent with the typical canonical form exhibited by the major OVAL content parsers.

Impact upon Existing OVAL Content

Any currently available *sql* and *sql57* OVAL content will remain valid due to the preservation of backward compatibility as explained above.

Impact upon Existing OVAL Implementations

Existing OVAL implementations would not only need to be extended with a new *sqlext* probe, but also undergo a refactoring of their target system handling algorithms to support the *instance* and *database* context behaviors. The current OVAL parser convention of characterizing scan targets as host operating system instances would need to be expanded to include support for a narrower scope that consists of individual software application and service instances.

Relevance to Existing OVAL Use Cases

Application of the *sqlext_test* schema most directly affects the Configuration Management OVAL use case. While the research for this proposal primarily considered Microsoft SQL Server and Oracle Database DISA STIG benchmarks, benchmarks against other DBMS engines such as MySQL and PostgreSQL are applicable as well.

Affected OVAL Schema Documents

The OVAL *sqlext* proposal extends XML schema documents *independent-definitions-schema.xsd* and *independent-system-characteristics-schema.xsd.*

Backward Compatibility with Previous Versions

The proposed *sqlext* OVAL test maintains backward compatibility by preserving the previous *sql* and *sql57* OVAL test schemata.

Compliance with Accepted Naming and Design Conventions

This proposal iterates upon the existing *sql* and *sql57* schemas and extends the standard OVAL *TestType* by incorporating known OVAL schema types that are already in use in many other currently available OVAL test schemata. Furthermore, the accompanying *SqlExtBehaviors* complex type is a standard reflection of the other independent definitions schema behavior complex types.

Potential Concerns Regarding this Proposal

This proposal deviates from the current OVAL processing model. The *sqlext_test* requires support for enumerating and iterating over SQL data stores at a global scope that transcends that of the actual probe instantiation. The *sqlext_test* also requires an implementation that performs a pre-scan interrogation of a given OVAL definition for any *instance* or *database* context behavior declarations in order to resolve the scope of the analysis. In the event that a narrower targeting scope is required, the given content must be processed multiple times against the same host system, once per each active DBMS instance or instance/database permutation as indicated by the *context* behavior.

Demonstration of the New Capabilities

The new *sqlext* OVAL capability has been demonstrated with a pre-release version of the SCAP Compliance Checker 4.1 and a sample SCAP benchmark based upon the Microsoft SQL Server 2012 DISA STIG.

Technical Overview of *sqlext* Specifications and Usage

The sqlext_*test* is used to check information stored in a database. This test has been designed to enable those settings to be tested. It extends the standard *TestType* as defined in the *oval-definitions-schema*. The required object element is a *sqlext*_object. An optional state element specifies the item field(s) to check.

The *sqlext*_object element is used by a *sqlext_test* to define the specific database and query to be evaluated. Connection information is specified by the *engine*, *version*, *instance*, and *database* string-type object entities. While entities *engine* and *version* declare which database management system software must be probed, *instance* and *database* entities describe the location of the dataset that must be queried. The most general application of the *sqlext* OVAL test would specify a ".*" wildcard as the *instance* entity value with operation *pattern match* while setting the *xsi:nil* property of the *database* entity as *true* in order to select the default database of all instances being hosted by a target DBMS. Alternatively, a content author that has the latitude to anticipate the exact names of target DBMS instances and database installations may instead declare the literal names as *instance* and *database* entity values with entity operation *equals*.

The *sqlext* object's default behavior (context='*host*') collects and evaluates all system characteristics from a given host system and produces a single result set. This arrangement may not be optimal for large DBMS deployments that host many instances and databases because the source of a failed evaluation will not be readily apparent in a single large consolidated result set. Therefore, the *sqlext_object* supports the *context* behavior, which provides the ability to produce separate results sets for each target (instance or database) on each host system. The *instance* context considers each DBMS instance as a separate scan target that generates its own result set. The *database* context handles each individual database as a separate scan target with a dedicated result set. By leveraging the more granular behaviors, content authors can greatly improve the clarity of collected results.

Fig. 1a: Example *sqlext_object*

```
<ind-def:sqlext_object id="oval:mil.disa.fso.mss2012:obj:2" version="1">
   <ind-def:engine var_ref="oval:mil.disa.fso.mss2012:var:1"/>
   <ind-def:version var_ref="oval:mil.disa.fso.mss2012:var:2"/>
   <ind-def:instance operation="pattern match">.*</ind-def:instance>
   <ind-def:database xsi:nil="true"/>
   <ind-def:sql>SELECT * FROM sys.server_permissions WHERE lower(state_desc) != 'grant'</ind-def:sql>
</ind-def:sqlext_object>
```

Fig. 1b: Default Content Processing Model
A single set of XML results per system, regardless of the number of DBMS instances.

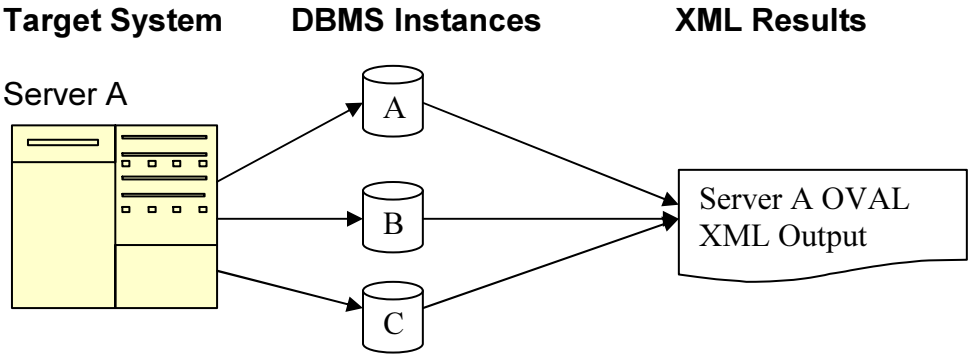
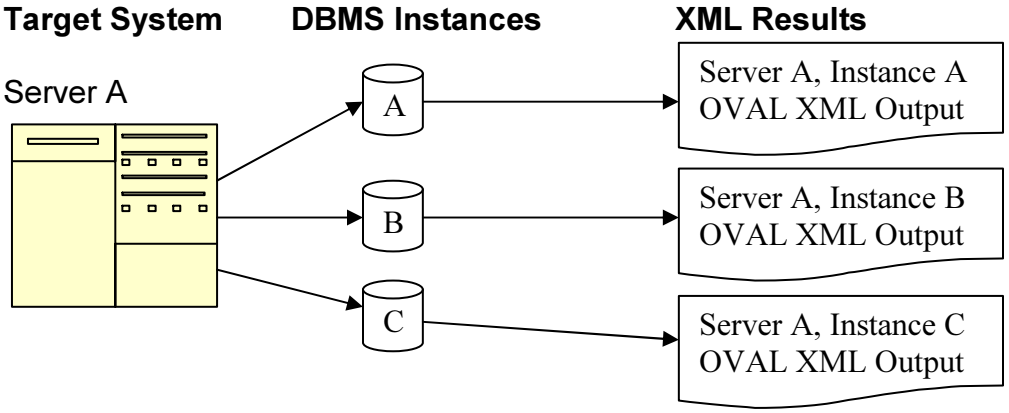


Fig. 2a: Example *sqlext_object*

```
<ind-def:sqlext_object id="oval:mil.disa.fso.mss2012:obj:2" version="1">
   <ind-def:behaviors context="instance"/>
   <ind-def:engine var_ref="oval:mil.disa.fso.mss2012:var:1"/>
   <ind-def:version var_ref="oval:mil.disa.fso.mss2012:var:2"/>
   <ind-def:instance operation="pattern match">.*</ind-def:instance>
   <ind-def:database xsi:nil="true"/>
   <ind-def:sql>SELECT * FROM sys.server_permissions WHERE lower(state_desc) != 'grant'</ind-def:sql>
</ind-def:sqlext_object>
```

Fig. 2b: *SQLEXT* Content Processing Model Where Context is "*instance*"
Separate sets of OVAL XML result files per database instance.



For best results, all *sqlext* tests should specify the same context, and the use of any other non-context-sensitive OVAL tests in the same stream should be minimal. When composing a SCAP stream with the *sqlext* OVAL schema, another recommended practice is to leverage *sqlext* as a CPE-OVAL applicability check. Instead of declaring a query as the *sql* object entity value, simply set its

xsi:nil property as *true*, which will determine the existence of the specified DBMS instances or named databases without attempting to query them. Then employ the same *sqlext context* behavior that is used in the primary OVAL definition in order to completely skip any non-applicable DBMS instances or named databases.

Fig. 3: Example Applicability Check for SQL Server 2012

```
<ind-def:sqlext_test id="oval:mil.disa.fso.mss2012:tst:1" version="1" check_existence="at_least_one_exists" check="all"
comment="This benchmark is applicable if at least one SQL Server 2012 instance is online.">
   <ind-def:object object_ref="oval:mil.disa.fso.mss2012:obj:1"/>
</ind-def:sqlext_test>

<ind-def:sqlext_object id="oval:mil.disa.fso.mss2012:obj:1" version="1">
   <ind-def:behaviors context="instance"/>
   <ind-def:engine>sqlserver</ind-def:engine>
   <ind-def:version>2012</ind-def:version>
   <ind-def:instance operation="pattern match">.*</ind-def:instance>
   <ind-def:database xsi:nil="true"/>
   <ind-def:sql xsi:nil="true"/>
</ind-def:sqlext_object>
```

Each object extends the standard *ObjectType* as defined in the *oval-definitions-schema*. Please refer to that description for more information. The common *set* element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the *oval-definitions-schema*.

Fig. 4: Example *sqlext_state* Evaluating a DB Table Field Named “count”

```
<ind-def:sqlext_state id="oval:mil.disa.fso.mss2012:ste:1" version="1">
   <ind-def:result>
      <oval-def:field name="count">0</oval-def:field>
   </ind-def:result>
</ind-def:sqlext_state>
```

Fig. 5: Example *sqlext_item*

```
<ind-sc:sqlext_item status="exists" id="3">
   <ind-sc:engine>sqlserver</ind-sc:engine>
   <ind-sc:version>2012</ind-sc:version>
   <ind-sc:instance>MSSQLSERVER</ind-sc:instance>
   <ind-sc:database></ind-sc:database>
   <ind-sc:sql>SELECT COUNT(credential_id) AS count FROM master.sys.master_key_passwords</ind-sc:sql>
   <ind-sc:result datatype="record">
      <oval-sc:field name="count">0</oval-sc:field>
   </ind-sc:result>
</ind-sc:sqlext_item>
```

Implications for XCCDF/ARF Output

While not technically in scope for the OVAL XML schema or the OVAL Developers list, it bears mentioning that the *sqlext instance* and *database* context behaviors support generating not only multiple OVAL XML result files, but also multiple associated XCCDF and SCAP ARF XML result files where applicable. In light of the fact that OVAL content is most commonly applied as the assessment automation layer of SCAP content streams, SPAWAR felt it relevant to make a note here, which would need to be addressed with the SCAP community in the event that this extension is officially folded into OVAL.