

## LOCKSS Difference Report

The LOCKSS Difference Report (report) compares thirteen items in the Schema File with their raw or computed counterparts in the CacheManager Database (CM). It is an XML file with a Summary section followed by four detail sections. The first three detail sections, one each for au's, hosts, and host-au combinations, display an "expected" value from the Schema file, an "observed" value from the CM, and a "diff" flag to indicate whether they match or not. The final detail section displays au's found in the CM but missing from Schema; this can be used to update the Schema to match the CM. The entries in the table below are used to drive the comparison routine and supply names for the XML elements.

The discussion that follows

- A. explains the table
- B. shows how items enumerated in the table are retrieved from the CM;
- C. explains "constructed" or "derived" items (measures);
- D. describes how the comparisons are done.

### Section A. Table explanation

The left column represents an "xpath" that points to a set of related elements or single element in the Schema. The xpath is either generic, meaning that it is replicated in the Schema for each host and au (items 1-9), or specific, meaning that it references a single element that occurs only once (items 10-13). The names of the "expected" elements in the report come from this column. The right column contains names of the "observed" elements in the report, most of which are CM "table.column" names or a close variant, except item 13. Names were chosen so one familiar with the Schema and CM can readily associate report elements with their source data.

Schema Items 1-9 bear a one-to-one correspondence to specific CM table columns. This means a single element value for a specific au or host listed in the Schema can be matched with a single table-column-row value (cell) for that au or host in the CM. For these items the report routine performs a direct comparison between the corresponding data elements.

Schema items 10-12 also correspond to specific CM table columns (we believe), but occur only once in the Schema (that is, they are constants that apply to all hosts or au's). The values of these items in the CM pertain to host-au combinations, not individual au's or hosts. Further, the comparisons of interest in these cases are between a Schema constant and value derived from a CM cell, not a raw CM cell value,

Schema item 13 is similar to 10-12 in that it is a constant, but differs in that it pertains to au's, not host-au combinations. It is constructed from the CM, but has no exact representation there so the right column is an identifier that doesn't reflect any particular table-column origin.

1. /SSP/hosts/host/@host_id	caches.ip_address
2. /SSP/hosts/host/hostIdentity/name	caches.name
3. /SSP/hosts/host/hostIdentity/accessBase/@adminEmail	platform_status.admin_email
4. /SSP/hosts/host/hostCapabilities/lockssVersion/@softwareVersion	caches.version
5. /SSP/hosts/host/hostCapabilities/storageAvailable/@max_size	disks.size
6. /SSP/archivalUnits/au/@au_id	archival_units.key_string
7. /SSP/archivalUnits/au/auIdentity/name	archival_units.name
8. /SSP/archivalUnits/au/auIdentity/accessBase/@accessURI	archival_units.base_url
9. /SSP/archivalUnits/au/auCapabilities/storageRequired/@max_size	cache_archival_units.size
10. /SSP/network/networkCapabilities/pollAgreement/@percent ( was host_au_agree )	cache_archival_units_status
11. /SSP/network/networkCapabilities/replicationFrequency/@maxHours ( was host_au_hours_since_last_crawl )	cache_archival_units_last_crawl
12. /SSP/network/networkCapabilities/verificationFrequency/@maxHours ( was host_au_hours_since_last_poll )	cache_archival_units_last_poll

## Section B. SQL used to retrieve the CM data

### Query 1 - SQL to retrieve items 6-8:

```
SELECT archival_units.id, archival_units.name, archival_units.base_url, archival_units.key_string FROM archival_units ;
```

This retrieval is self-explanatory.

### Query 2 - SQL to retrieve item 9 and derive item 10:

```
SELECT DISTINCT cache_archival_units.id, cache_archival_units.archival_unit_id,  
cache_archival_units.status, cache_archival_units.size, cache_archival_units.key_string  
FROM cache_archival_units WHERE ( status LIKE "%100%" ) GROUP BY archival_unit_id ORDER BY NULL ;
```

The 'cache\_archival\_units' table contains a row for each host-au combination where a host actually has a copy of an au, not all possible host-au combinations. There are currently 9 x 28 (252) possible host-au combinations, but only 89 rows in the table. We decided we would trust a size figure only if polling indicated 100% agreement across hosts, thus the WHERE statement. Then, since we want only one size figure, we specify DISTINCT, to keep only the first row that matches. Note that we do not check that other rows with 100% agreement (same au but another host) have the same size figure as the first row. With the limited data in the database this cannot be tested because it never occurs. It is possible though, and we should decide how it would be handled.

### Query 3 - SQL to retrieve items 1-5

```
SELECT caches.id, caches.ip_address, caches.name, caches.version, caches.hostname,  
disks.cache_id, disks.size, platform_status.cache_id, platform_status.admin_email  
FROM caches  
LEFT JOIN platform_status ON caches.id=platform_status.cache_id  
LEFT JOIN disks ON platform_status.cache_id=disks.cache_id;
```

While this retrieval returns reasonable result, I am not completely confident that the double LEFT JOIN is the proper approach. Note that the 'caches' table always contain one row for each cache/host regardless of the host status, but the 'disks' and 'platform\_status' tables have only have rows for hosts that are up. Thus, any retrieval seeking a complete enumeration of hosts should include the 'caches' table.

### Query 4 - SQL to retrieve data to derive aggregate/summary items 11-13.

```
SELECT cache_archival_units.key_string, cache_archival_units.status, caches.ip_address,  
cache_archival_units.last_poll, cache_archival_units.last_crawl  
FROM cache_archival_units, caches  
WHERE cache_archival_units.cache_id=caches.id  
ORDER BY cache_archival_units.key_string;
```

This returns a table where each row corresponds to a host-au combination where a host actually has a copy of an au (not all possible combinations).

