Table of Contents

============================== Working with use cases in HITS ==============================

There are several assurance use cases supported by HITS. All involve SIF CRUD operations over REST, and can confirm your ability to provide a hub provisioning data through SIF.

All the use cases follow a similar pattern. The following is our suggestion as to how to proceed through them:

**1.  Study the use case:**

The use case lists the kinds of objects involved in the exchange, the anticipated choreographies, and how the objects will be validated. [Here is an example usecase...](/docs/basic_daily_attendance/basic_daily_attendance.md)

**2.  Set up a client to use HITS.**

 For assistance with setting up a client, please [contact the NSIP team](mailto:info@nsip.edu.au). They will supply you with a personalised and data-populated environment for testing, accessed via a URL, which takes you to your personalised dashboard (example pictured below):

![](data:image/png;charset=UTF-8;base64,)

When a client is set up, an instance of the HITS database is populated just for you. HITS acts as a proxy for the hub you will be integrating into. Your database is populated with randomised objects that are representative of the information stored on a hub. To simplify testing, your client is assigned to one dummy school: all objects on the database are associated with that school.

![](data:image/png;charset=UTF-8;base64,)

![](data:image/png;charset=UTF-8;base64,)

**3.  Get some objects from HITS**

 The use case describes the objects from the hub that you need to access; we expect that you will get all objects of the given classes that are available from HITS. Each object class is associated with a distinct endpoint, named by appending "s" to the object class; you fetch instances of ScheduledActivity, for example, from the endpoint ending in /ScheduledActivitys.

![](data:image/png;charset=UTF-8;base64,)

![](data:image/png;charset=UTF-8;base64,)

![](data:image/png;charset=UTF-8;base64,)

![](data:image/png;charset=UTF-8;base64,)

**4.  Based on those objects, generate some new objects**

 Your client will be validated based on your ability to generate valid SIF objects and post them back to the hub. One of the main tests applied to your objects is their referential integrity: they need to link to objects that are already on the hub. A class roster that you generate, for example, needs to reference the existing students, staff, rooms, and subjects in the school. In order to create those links, you will need to access all the relevant objects from HITS (as noted above), and use only their GUIDs in generating the objects that reference them.

**5.  Clear the HITS database**

 The validation of your objects runs over the entire contents of your database instance. This means you can validate results only after you have posted all the objects you need to—which may involve several messages. If you have posted some objects with errors, you will need to clear the database before resending the corrected objects. This allows you to start with a clean slate, without having to worry about updating or deleting individual objects you have already posted: your rerun can be treated as new Creates on the endpoint.

**6.  Post those objects to HITS (*Return Path use cases*)**

Having created new objects, you will need to post them to the HITS endpoint specific to your client. If your post is successful, this will add the objects to the client database instance. As there is a different endpoint for each object class, you may need to post multiple messages. (On the other hand, SIF 3 allows you to package multiple objects of the same class into a single message.)

**NOTE 1:**

If the message is not well-formed XML, the message will be rejected:

![](data:image/png;charset=UTF-8;base64,)

![](data:image/png;charset=UTF-8;base64,)

**NOTE 2:**

The message will NOT be rejected if the message is well-formed, but does not follow the SIF-AU schema. HITS does not currently perform any syntactic validation of payloads against the SIF XML schema. This includes validating for failure to use the code sets prescribed in the SIF-AU specification. However, the payload populated may have empty elements where those elements have not been provided properly in the payload.

**7.  Run report on the objects you have posted**

Validation is run when:

* you have posted all the messages required by the use case to the object end points, and
* the messages have been successfully processed, and
* the resulting objects have been added to the client database.

 Validation runs over the objects added to the database, and checks that the objects make sense in the context of the use case. As a result of validation, a report is generated which indicates whether your objects satisfy the use case requirements:

![](data:image/png;charset=UTF-8;base64,)

The main things checked for are:

1. **Referential integrity**: your objects only link to the RefIDs of other objects on the database—whether these are objects that were already on the database (and that you have read from HITS), or objects that you have posted to the database in another message. Because you may need to spread your objects over many messages, do not run validation until you have posted all your messages.
2. **Object Obligations**: you have posted at least the number of instances of each object that is set in the use case. For example, a timetabling application cannot be validated unless it has posted at least one TimeTableCell object.
3. **Element Obligations**: you have posted all the elements in your object that are mandatory in the use case. This includes not only the elements that are mandatory in the SIF-AU schema, but also the elements that this use case specifically requires to be populated.
4. **Element Values**: some use cases will require that a particular value appear in an element. The objects will be tested to confirm that this has been done.