**10 Best Practices for xAPI Statements**

**10. Check for typos & duplicates**

We run a spell check on our text when we write for humans to read, so why not run some quality checks on the statement data that will need to be accurate for querying later? It’s important to use the correct identifiers for the Verbs and Activity Types in statements. Not only should we check for typos and validate the JSON, but it’s also important not to reinvent existing Verbs or Activity Types when many can be [reused from ADL](http://xapi.vocab.pub/).

***The Benefit:****A quality assurance check on your statements will result in more accurate data being stored in the Learning Record Store (LRS), which will ultimately result in less errors when trying to query over the data.*

**9. Reuse the requirements & templates from existing xAPI profiles**

Often designers get started quickly with xAPI and don’t realize there’s already a profile that defines a way to track the type of learning activity or interaction they want to track. A profile is the documentation used when implementing xAPI in a particular context. Several communities of practice have published profiles over the years for you to reuse. Need to track various types of video interactions? Check out the templates available in [the Video Profile](https://liveaspankaj.gitbooks.io/xapi-video-profile/content/). Need to track common eLearning interactions or send xAPI data from your existing SCORM content to an LRS? Check out [the SCORM Profile](https://adl.gitbooks.io/scorm-profile-xapi/content/). Many existing profiles have specific requirements and statement examples you can follow without having to spend time reinventing the wheel.

***The Benefit:****Several LRS applications have out-of-the-box dashboards and reports that are based on existing ways of storing xAPI data. Save time and get immediate gratification from seeing your LRS render useful analytics without having to build custom reports and queries.*

**8. Use context extensions to store additional information about the context of the learning environment**

The basic xAPI statement is very useful for storing general information about learning activities and outcomes. Many learning experience designers don't realize that additional contextual information can also be stored by using context extensions. This type of information can include things like competencies associated with satisfying the activity, the physical location of the learner, the organization or department they belong to, or even the type of browser, user interface, or other technology they are engaging in as part of the learning. An example of the JSON for this part of an xAPI statement is provided below. (Figure 2)

***The Benefit:****When additional contextual data is mixed with learning experience data, you have more opportunities for understanding additional characteristics about the learner or learning environment, which can lead to formulating predictive analytics.*

|  |
| --- |
| {   "context": {    "platform": "Moodle 3.8.3",    "extensions": {      "https://w3id.org/xapi/acme/extensions/department": "Accounting",      "https://w3id.org/xapi/acme/extensions/training-location":                ?    "Houston, TX",      "https://credreg.net/ctdlasn/terms/Competency#alignTo":                    "https://credentialengineregistry.org/resources/ce-4451c5bf-6e1f-4935-9e63-e1c4f18b6e18",      "https://w3id.org/xapi/acme/extensions/os-browser":  "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36"     }    }  } |

***Figure 2*: Context extension**

**7. Use result and result extensions to store response and outcome information**

I’ve seen various forms of results or outcome data stored in really odd places (i.e., context extensions) in xAPI statements over the years because designers didn’t initially realize there was already a place to store that type of information. The xAPI statement data model has a “bucket” available for storing user responses to interactions, learning activities, and their collective outcomes. This result bucket is most commonly used for storing learning or performance results such as completion, pass/fail, scores, and duration. Many learning experience designers may not know that they can also store how users respond or interact with things and the associated result of that response or interaction. If you want to track other types of results beyond what is typically associated with learning (e.g., completion, scores, pass/fail, etc), then you can also use result extensions to support additional strategies for measuring outcomes. An example of the JSON for this part of an xAPI statement is provided below. (Figure 3)

***The Benefit:****Storing results and outcome data are critical for most analytics. Leveraging the result and result extensions parts of the xAPI statement will lead to more easily querying and creating reports on your data because you’ll know exactly where to look and so will the LRS.*

|  |
| --- |
| {    "Result":{       "score": {          "scaled": 0.90       },       "completion": "true",       "success": "true",?     "duration": "PT1H10M5S",       "extensions":{         "https://w3id.org/xapi/acme/extensions/progress": 1.0,         "https://w3id.org/xapi/acme/extensions/accounting-kpi": "payroll-processing"      }    }  } |

***Figure 3*: Result extension**

**6. Use appropriate Verbs to indicate the start and end of a session**

This practice isn’t just for learning experience designers and developers implementing courses. Most learning experiences (not just courses) are recorded linearly through time. A communication session ties together all the data generated as a learner is experiencing the content, starting once the content has been initialized (or resumed) and ending when the content has been terminated (or suspended). Too often learning designers only use very basic Verbs throughout the session such as “attempted” or “experienced” or “progressed” and don’t provide appropriate Verbs to identify the beginning or end. It’s a best practice to always use Verbs such as “initialized” to identify the beginning of a session and “terminated” to identify the end of a session and include a context.registration identifier to tie together an attempt in a session (see #2). The “terminated’ statement can also include the final outcome information discussed in #5.

***The Benefit:****Strategically identifying these points in a learning experience is important for advanced analytics such as showing how the user progresses through different learning paths. If you have a need to show this type of learning path progression, then you’ll benefit from adding relevant Verbs to identify the key points in a session.*

**5. Define Activities by their type**

What type of Activity did the learner interact with? Was it a lesson, course, mobile app, question, assessment, eBook, simulation, or even a video? If you want to build any useful queries and analytics over your xAPI data without much extra work, then defining the type of Activity upfront in your statements is a must. You can simply define this in the statement data model by setting the object.definition.type to the identifier associated with the relevant Activity Type. [ADL’s xAPI Vocabulary listing](http://xapi.vocab.pub/activity-types/index.html) (and future Profile Server) offers a plethora of Activity Types available for you to choose from and use to define the “thing” being tracked by your learner. An example of the JSON for this part of an xAPI statement is highlighted below. (Figure 4)

***The Benefit:****Defining the Activity type critical for most analytics and will lead to more easily querying and creating reports on your Activity data because you’ll know exactly what type of Activity your analytics are targeting. Most LRSs will also provide out-of-the-box reports and charts that support the most common Activity Types.*

|  |
| --- |
| {   "object":{    "id": "https://acme.com/xapi/activities/lessons/3e32f474-af07-21ea-b3de-0242ac130003",    "objectType": "Activity",    "definition": {        "name": {        "en": "Payroll Processing"        },        "description": {        "en": "An instructional lesson on payroll processing associated with the accounting curriculum at Acme,Inc."        },        "type": "<http://adlnet.gov/expapi/activities/lesson>"      }    }  } |

***Figure 4*: Activity type**

**4. Create unique identifiers for each Activity**

Probably one of the most important things that can be done to your xAPI Statements is to ensure that there is absolutely no possibility of accidentally creating or using the same Activity IDs for different activities or creating a new unique ID for each session. In other words, the Activity ID (aka object ID) must be a globally unique identifier for each thing you are tracking in xAPI. The identifier for the Activity should not be a real URL or path to a file. It’s simply just an identifier. A common mistake that I’ve seen (and made myself) is to use the browser’s window.location.href value as the Activity ID. This is a bad idea because the content might be moved around and accessed from different servers. If that happens then you would end up with multiple identifiers for the same Activity. It’s a great idea to come up with a consistent naming convention or scheme for your identifiers and make sure the same identifier is never reused for different Activities. An example of the JSON for this part of an xAPI Statement is highlighted below. (Figure 5)

***The Benefit:****If two different Activities have the same identifier, then learners could accidentally get credit for the wrong learning experience. That’s a pretty big deal! Creating unique Activity IDs is extremely important for the veracity of your xAPI datal!*

|  |
| --- |
| {   "object":{    "id": "https://acme.com/xapi/activities/lessons/3e32f474-af07-21ea-b3de-0242ac130003",    "objectType": "Activity",    "definition": {        "name": {        "en": "Payroll Processing Lesson"        },        "description": {        "en": "An instructional lesson on payroll processing associated with the accounting curriculum at Acme,Inc."        },        "type": "<http://adlnet.gov/expapi/activities/lesson>"      }    }  } |

***Figure 5*: Activity ID**

**3. Represent common assessment interactions according to the xAPI Spec**

The xAPI specification provides a standard way to track assessment data with built-in definitions for computer managed instruction (CMI) interactions. These interaction types were borrowed from SCORM and are intended to provide a familiar way for recording assessment interaction data. The questions (CMI interaction Activities) should have a relationship to the parent assessment Activity (see #1. Use Parent and Grouping ContextActivities to represent relationships between Activities) below. The answer to each assessment question interaction should also be stored in result.response. Don’t reinvent the wheel. Make sure your statements are at least modeled like [the examples in the Appendix in the xAPI Spec](https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Data.md#Appendix2C). (See Figure 6)

***The Benefit:****Many LRSs have analytics platforms with dashboards that automatically render based on assessment data being modeled this way. You won’t have to create custom reports and dashboards in some cases if you follow the standard way assessment data is modeled by using CMI interactions.*

|  |
| --- |
| {    "object": {        "id":  "https://acme.com/xapi/activities/cmi.interactions/57213bb2-b1f7-420e-9042-952bde6fe72b",        "definition": {        "name": {        "en": "Question 1"        },        "description": {        "en": "All states have set their minimum wage to be the same as the federal government."        },        "type": "http://adlnet.gov/expapi/activities/cmi.interaction",        "interactionType": "true-false"        }    },    "result": {        "response": "false"    }  } |

***Figure 6*: Assessment interaction**

**2. Use a unique context.registration identifier in all statements with an attempt**

The context.registration property should be used any time you want to identify multiple xAPI Statements that are all part of a particular attempt. An attempt on an Activity may span one or more communication sessions and the attempt may be suspended and resumed. For example, recording the number of attempts on a lesson in a course is possible in xAPI by grouping together Statements with a unique Registration identifier. Measuring the attempts on Activities is useful for more than just lessons or courses and can be implemented in any type of xAPI project. An example of the JSON for this part of an xAPI Statement is highlighted below. (Figure 7)

***The Benefit:****This practice makes it possible for reports and analytics to identify the Statements that should be considered part of a single attempt.*

|  |
| --- |
| {   "context": {    "registration": "a7bbe8a0-6c77-41a0-a230-9f70d3df9204",    "platform": "Moodle 3.8.3",    "extensions": {      "https://w3id.org/xapi/acme/extensions/department": "Accounting",      "https://w3id.org/xapi/acme/extensions/training-location":                ?    "Houston, TX",      "https://credreg.net/ctdlasn/terms/Competency#alignTo":                    "https://credentialengineregistry.org/resources/ce-4451c5bf-6e1f-4935-9e63-e1c4f18b6e18",      "https://w3id.org/xapi/acme/extensions/os-browser":  "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36"    }   }  } |

***Figure 7*: Context registration identifier**

**1. Use Parent and Grouping Context Activities to represent relationships between Activities**

I’ve often seen xAPI Statements that have the same value for the grouping and parent Activities, which can cause problems with generating meaningful analytics. Activities can be linked together to express both indirect and direct relationships, but they should never be the same value. Granular xAPI activities have an indirect grouping relationship with a broader Activity. For example, a question (cmi.interaction) in a test should have a grouping relationship with the course. Activities can also be grouped together to express direct relationships between a particular child Activity and a single parent. For example, a question (cmi.interaction) in an assessment should have a parent relationship with the assessment. An example of the JSON for some of these key parts of an xAPI Statement with both grouping and parent relationships are highlighted below. (Figure 8)

***The Benefit:****Establishing relationships by using Context Activities provides a way to establish the levels of granularity in between your content and will result in more robust reporting and analytics. It will also help with aggregating granular analytics across parent activities and enables exploratory analytics by drilling down or up into related activities.*