1 xAPI Formal Specification

The current formal specification only defines xAPI statements abstractly within the context of Z. A concrete definition for xAPI statements is outside the scope of this document.

1.1 Basic and Free Types

$[MBOX, MBOX\_SHA1SUM, OPENID, ACCOUNT]$

- Basic Types for the abstract representation of the different forms of Inverse Functional Identifiers found in xAPI

$[CHOICES, SCALE, SOURCE, TARGET, STEPS]$

- Basic Types for the abstract representation of the different forms of Interaction Components found in xAPI

$IFI ::= MBOX \mid MBOX\_SHA1SUM \mid OPENID \mid ACCOUNT$

- Free Type unique to Agents and Groups, The concrete definition of the listed Basic Types is outside the scope of this specification

$OBJECTTYPE ::= Agent \mid Group \mid SubStatement \mid StatementRef \mid Activity$

- A type which can be present in all activities as defined by the xAPI specification

$INTERACTIONTYPE ::= true-false \mid choice \mid fill-in \mid long-fill-in \mid matching \mid performance \mid sequencing \mid likert \mid numeric \mid other$

- A type which represents the possible interactionTypes as defined within the xAPI specification

$INTERACTIONCOMPONENT ::= CHOICES \mid SCALE \mid SOURCE \mid TARGET \mid STEPS$

- A type which represents the possible interaction components as defined within the xAPI specification
- the concrete definition of the listed Basic Types is outside the scope of this specification

$CONTEXTTYPES ::= parent \mid grouping \mid category \mid other$

- A type which represents the possible context types as defined within the xAPI specification

$[STATEMENT]$

- Basic type for an xAPI data point

$[AGENT, GROUP]$

- Basic types for Agents and collections of Agents

1.2 Id Schema

Id

$id : \mathbb{F}_1 \#1$

- the schema Id introduces the component id which is a non-empty, finite set of 1 value

1.3 Schemas for Agents, Groups and Actors

$Agent$

$agent : AGENT$
$objectType : OBJECTTYPE$
$name : \mathbb{F}_1 \#1$
$ifi : IFI$

$objectType = Agent$
$agent = \{ifi\} \cup \mathbb{P}\{name, objectType\}$

- The schema $Agent$ introduces the component $agent$ which is a set consisting of an ifi and optionally an $objectType$ and/or $name$

$Member$

$Agent$
$member : \mathbb{F}_1$

$member = \{a : AGENT \mid \forall a_n : a_i..a_j \bullet i \leq n \leq j \bullet a = agent\}$

- The schema $Member$ introduces the component $member$ which is a set of objects a, where for every a within $a_0..a_n$, a is an $agent$

$Group$

$Member$
$group : GROUP$
$objectType : OBJECTTYPE$
$ifi : IFI$
$name : \mathbb{F}_1 \#1$

$objectType = Group$
$group = \{objectType, name, member\} \lor \{objectType, member\} \lor$
$\{objectType, ifi\} \cup \mathbb{P}\{name, member\}$

- The schema $Group$ introduces the component $group$ which is of type $GROUP$ and is a set of either $objectType$ and $member$ with optionaly $name$ or $objectType$ and ifi with optionally $name$ and/or $member$

Actor

$Agent$
$Group$
$actor : AGENT \lor GROUP$

$actor = agent \lor group$

- The schema *Actor* introduces the component *actor* which is either an *agent* or *group*

1.4 Verb Schema

Verb

Id
$display, verb : \mathbb{F}_1$

$verb = \{id, display\} \lor \{id\}$

- The schema $Verb$ introduces the component *verb* which is a set that consists of either *id* and the non-empty, finite set *display* or just *id*

1.5 Object Schema

Extensions

$extensions, extensionVal : \mathbb{F}_1$
$extensionId : \mathbb{F}_1 \# 1$

$$extensions = \{e : (extensionId, extensionVal) \mid \forall e_n : e_i..e_j \bullet i \leq n \leq j \bullet (extensionId_i, extensionVal_i) \lor (extensionId_i, extensionVal_j) \land (extensionId_j, extensionVal_i) \lor (extensionId_j, extensionVal_j) \land extensionId_i \neq extensionId_j\}$$

- The schema *Extensions* introduces the component *extensions* which is a non-empty, finite set that consists of ordered pairs of $extensionId$ and $extensionVal$. Different $extensionId$s can have the same $extensionVal$ but there can not be two identical $extensionId$ values
- $extensionId$ is a non-empty, finite set with one value
- $extensionVal$ is a non-empty, finite set

InteractionActivity

$interactionType : INTERACTIONTYPE$
$correctResponsePattern : \text{seq}_1$
$interactionComponent : INTERACTIONCOMPONENT$

$$interactionActivity = \{interactionType, correctReponsePattern, interactionComponent\} \lor \{interactionType, correctResponsePattern\}$$

- The schema $InteractionActivity$ introduces the component $interactionActivity$ which is a set of either $interactionType$ and $correctResponsePattern$ or $interactionType$ and $correctResponsePattern$ and $interactionComponent$

$$
\begin{array}{|l}
\hline
Definition \\
\hline
InteractionActivity \\
Extensions \\
definition, name, description : \mathbb{F}_1 \\
type, moreInfo : \mathbb{F}_1 \# 1 \\
\hline
definition = \mathbb{P}_1\{name, description, type, moreInfo, extensions, interactionActivity\} \\
\hline
\end{array}
$$

- The schema $Definition$ introduces the component $definition$ which is the non-empty, finite power set of $name$, $description$, $type$, $moreInfo$ and $extensions$

$$
\begin{array}{|l}
\hline
Object \\
\hline
Id \\
Definition \\
Agent \\
Group \\
Statement \\
objectTypeA, objectTypeS, objectTypeSub, objectType : OBJECTTYPE \\
substatement : STATEMENT \\
object : \mathbb{F}_1 \\
\hline
substatement = statement \\
objectTypeA = Activity \\
objectTypeS = StatementRef \\
objectTypeSub = SubStatement \\
objectType = objectTypeA \lor objectTypeS \\
object = \{id\} \lor \{id, objectType\} \lor \{id, objectTypeA, definition\} \\
\qquad \lor \{id, definition\} \lor \{agent\} \lor \{group\} \lor \{objectTypeSub, substatement\} \\
\qquad \lor \{id, objectTypeA\} \\
\hline
\end{array}
$$

- The schema $Object$ introduces the component $object$ which is a non-empty, finite set of either id, id and $objectType$, id and $objectTypeA$, id and $objectTypeA$ and $definition$, $agent$, $group$, or $substatement$
- The schema $Statement$ and the corresponding component $statement$ will be defined later on in this specification

1.6 Result Schema

Score

$$score : \mathbb{F}_1$$
$$scaled, min, max, raw : \mathbb{Z}$$

$$scaled = \{n : \mathbb{Z} \mid -1.0 \leq n \leq 1.0\}$$
$$min = n < max$$
$$max = n > min$$
$$raw = \{n : \mathbb{Z} \mid min \leq n \leq max\}$$
$$score = \mathbb{P}_1\{scaled, raw, min, max\}$$

- The schema *Score* introduces the component *score* which is the non-empty powerset of *min*, *max*, *raw* and *scaled*

Result

$$Score$$
$$Extensions$$
$$success, completion, response, duration : \mathbb{F}_1 \#1$$
$$result : \mathbb{F}_1$$

$$success = \{true\} \lor \{false\}$$
$$completion = \{true\} \lor \{false\}$$
$$result = \mathbb{P}_1\{score, success, completion, response, duration, extensions\}$$

- The schema *Result* introduces the component *result* which is the non-empty power set of *score*, *success*, *completion*, *response*, *duration* and *extensions*

1.7 Context Schema

Instructor

$$Agent$$
$$Group$$
$$instructor : AGENT \lor GROUP$$

$$instructor = agent \lor group$$

- The schema *Instructor* introduces the component *instructor* which can be either an *agent* or a *group*

$Team$

$Group$
$team : GROUP$

$team = group$

- The schema $Team$ introduces the component $team$ which is a $group$

$Context$

$Instructor$
$Team$
$Object$
$Extensions$
$registration, revision, platform, language : \mathbb{F}_1 \#1$
$parentT, groupingT, categoryT, otherT : CONTEXTTYPES$
$contextActivities, statement : \mathbb{F}_1$

$statement = object \setminus (id, objectType, agent, group, definition)$
$parentT = parent$
$groupingT = grouping$
$categoryT = category$
$otherT = other$
$contextActivity = \{ca : object \setminus (agent, group, objectType, objectTypeSub, substatement)\}$
$contextActivityParent = (parentT, contextActivity)$
$contextActivityCategory = (categoryT, contextActivity)$
$contextActivityGrouping = (groupingT, contextActivity)$
$contextActivityOther = (otherT, contextActivity)$
$contextActivities = \mathbb{P}_1\{contextActivityParent, contextActivityCategory, contextActivityGrouping, contextActivityOther\}$
$context = \mathbb{P}_1\{registration, instructor, team, contextActivities, revision, platform, language, statement, extensions\}$

- The schema $Context$ introduces the component $context$ which is the non-empty powerset of $registration$, $instructor$, $team$, $contextActivities$, $revision$, $platform$, $language$, $statement$ and $extensions$
- The notation $object \setminus agent$ represents the component $object$ except for its subcomponent $agent$

1.8 Timestamp and Stored Schema

$Timestamp$

$timestamp : \mathbb{F}_1 \#1$

$Stored$

$stored : \mathbb{F}_1 \#1$

- The schema $Timestamp$ and *stored* introduce the components *timestamp* and *stored* respectively. Each are non-empty, finite sets containing one value

1.9 Attachments Schema

Attachments

$$display, description, attachment, attachments : \mathbb{F}_1$$
$$usageType, sha2, fileUrl, contentType : \mathbb{F}_1 \#1$$
$$length : \mathbb{N}$$

$$attachment = \{usageType, display, contentType, length, sha2\} \cup \mathbb{P}\{description, fileUrl\}$$
$$attachments = \{a : attachment\}$$

- The schema *Attachments* introduces the component *attachments* which is a non-empty, finite set of the component *attachment*
- The component *attachment* is a non-empty, finite set of the components $usageType$, *display*, $contentType$, *length*, *sha*2 with optionally *description* and/or $fileUrl$

1.10 Statement and Statements Schema

Statement

$$Id$$
$$Actor$$
$$Verb$$
$$Object$$
$$Result$$
$$Context$$
$$Timestamp$$
$$Stored$$
$$Attachments$$
$$statement : STATEMENT$$

$$statement = \{actor, verb, object, stored\} \cup \mathbb{P}\{\mathrm{id}, result, context, timestamp, attachments\}$$

- The schema *Statement* introduces the component *statement* which consists of the components *actor*, *verb*, *object* and *stored* and the optional components *id*, *result*, *context*, *timestamp*, and/or *attachments*
- The schema *Statement* allows for subcomponent of *statement* to referenced via the . (selection) operator

$$
\begin{array}{|l}
\hline
\textit{Statements} \\
\textit{Statement} \\
\textit{IsoToUnix} \\
\textit{statements} : \mathbb{F}_1 \\
\hline
\textit{statements} = \{s : \textit{statement} \mid \forall s_n : s_i..s_j \bullet i \leq n \leq j \\
\qquad \bullet \textit{convert}(s_i.\textit{timestamp}) \leq \textit{convert}(s_j.\textit{timestamp})\} \\
\hline
\end{array}
$$

- The schema *Statements* introduces the component *statements* which is a non-empty, finite set of the component *statement* which are in chronological order.