The Items definition (see Items Description for Runtime Engine.doc) defines all the items as well as any statically defined times at which items are sent.

From the runtime’s point of view, it should parse the XML file and get two collections:

1. A collection of “ScheduledItem” objects, which are Items to be sent at a given time
2. A description of each item

1- For the timestamped collection, this should correlate a time (in seconds from start of the scenario) with an item to be played at that time (reference to an Item’s “ID” field).

2 – Each Item contains an ID as well as some parameters and an Action section. An Item can be defined with an empty Action section, or with some set of Move, Reveal, Interaction, or StateChange actions. The runtime engine, when choosing an item from the ScheduledItem collection, should check to see if any specific actions are defined. If no actions are defined, then the Runtime Engine should have some logic to create some Actions appropriate to that item (i.e. based on the Parameters of Crossing, Groupings, Threat, Threat Type, PlayerResources, TeammateResources).

These parameters are defined as:

* Crossing – True/False – If true, then there should be an asset which crosses areas of responsibility from one player to another, requiring and hand-off. A crossing might start with an object being placed and then moving to cross.
* Grouping – One/Two – One grouping is one or more objects appearing WITHIN two grid cells (approx. 25km) of each other. A two grouping is two or more objects that are AT LEAST two grid cells apart.
* Ambiguity – Ambiguous/Unambiguous–if it’s ambiguous then it should not be obviously identifiable, no IFF, not attacking, not pursuing another object. To make something obviously identifiable, it can be Revealed with the these behaviors.
* Threat Type – Threat/Non-threat
* PlayerResources – Available/Unavailable – If “available”, then the Item location should be near the Item’s player’s resources (If BAMS, near BAMS, else near Firescouts).   
  NOTE: It’s tough to specify that a user has Tracks available, so by placing an object near a player’s resources or away from their resources we might be able to control how they can interact with the object.
* TeammateResources – Available/Unavailable – Similar to the previous section, if “available” then the item should be placed near the other teammate’s resources. There’s an interesting case when both PlayerResources and TeammateResources are either Available or Unavailable. Not sure how Courtney would want this handled.

Ambiguity means randomness for all assets

Unambiguity means displaying ground truth for all assets