TIER Event-Driven Messaging (EDM) Architecture

Collection of Bullets

1. Requirements analysis (Lots of questions, a few answers)
    a. Message carries identifier for resource (person, group, etc.) that has changed.
        i. Does the payload of the message contain a (complete or partial) resource representation as of the time when the message was generated?
        ii. If a message says ‘group H changed’, the recipient would have to pull the full membership of the group to see what changed…
        iii. Order matters. How do you support that in an async system
        iv. We may not want cloud services to do queries back (or the service may not be able to do it, so all the resource info has to be passed
        v. Identifier only preserves loose coupling
        vi. Or, here's the info, and here’s a link to it
    b. Are the events finer grained than ‘something changed” about this resource’?; Event types like add, modify...
        i. E.g. a message from Grouper could be: Add/delete member; that kind of minimal info should be passed in the message
        ii. Should the granularity be as fine as a specific attribute-value change to a specific resource?
        iii. Should the granularity as coarse as an abstract “business event” and a bag of metadata along with it?
    c. Message posted on a named channel that is available for opt-in subscription by message consuming endpoints.
    d. Message consumer is responsible for action that follow receipt of the message, for example, it might retrieve a representation of the changed resource, likely via Restful API.
    e. Message payloads must be protected in transit against interception and eavesdropping.
    f. Consider using the Event-sourcing model [1], [2]
    g. Think in terms of delivering a Minimum Viable Product:
        i. EDM with a design that solves the most common problems. Allow others to add their own pieces to solve your specific problem.
        ii. Integrating ‘dumb’ cloud services with EDM: Local service wrapper subscribes to messages, does the query for resource representation and then invokes provisioning APIs on the ‘dumb’ cloud service.
    h. Message posted on a named channel that is available for opt-in subscription by message consuming endpoints. Open issue.
    i. Message consumer is responsible for action that follow receipt of the message, for example, it might retrieve a representation of the changed resource, likely via Restful API, or make changes to systems based on information received in the message.
    j. Message payloads must be protected in transit against interception and eavesdropping.
    k. Consider using the Event-sourcing model [1], [2]

2. Start an EDM FAQ with questions we come up with and answers provided by the ‘experts’ among us or from outside

- Issues with distributed asynchronous environments generally
    - Q: In a multi-component distributed environment, how does the system attain a consistent state in the presence of indeterminate timing of events across components.
- Issues specifically related to messaging protocols
    - Q: How do we support interoperability between messaging systems? Messaging protocols are somewhat like Relational databases, there are many flavors of SQL, each package seems to have its own.