RCP - WEBAPI-027 Event Resource and Replication Model

Submitter Name	Josh Darnell
Submitter Organization	RESO
Submitter Email	josh@reso.org
Co-submitter Name	Paul Stusiak
Co-submitter Organization	Falcon Technologies Corp
Co-submitter Email	pstusiak@falcontechnologies.com

Document Name	Web API
Document Version	1.2
Date Submitted	2019-04-01
Status	APPROVED
Status Change Date	

Synopsis

A new standard Resource and method for replication is described.

Rationale

One of the most common business cases for real estate data is the replication of listing and other data between producer and consumer sources. The current state of the art is to use modification timestamps, long polling and state inference to synchronize data between two servers. The current scheme places an additional burden on the consumer of data to know the state of the producer and coordinate timestamps across vendors. This results in a need to frequently resynchronize the data on a because it is difficult or impossible to resolve timestamps or record state. This proposal addresses some of the problems with the current scheme by replacing timestamps with an event log and providing a Odata Resource that exposes the event log.

Proposal

A new Odata Resource will be created, as mentioned in the corresponding Data Dictionary proposal. This Resource will provide a logical timestamp of events and the Resource and Entity that the event affected. The logical timestamp is an event identifier that denotes that a business event occurred. Business events are rules or actions that represent the business logic of the resources of the system. Examples of business events could be a listing price change, a listing status change, an phone number change for an agent and the addition or deletion of a photo to any other resource that has media object. These business events and the associated logic exists in current systems implementing the Web API today. By representing these events in a consistent way, eventual record state consistency can be achieved.

A new well-known name, ***EventID*** will be added. ***EventID*** is part of an event record that describes all events for all resource in order. This event record is a compact representation of an event that occurred. ***EventID*** is a durable, immutable, monotonic identifier that preserves the order that events occur in a system. It can only increase in value. A value, a, for ***EventID*** that is arithmetically less than another value, b, for ***EventID*** is defined such that event a occurs earlier in time than event b. For example, An ***EventID*** EventID=200 (a), compared with another ***EventID*** EventID=1001 (b), satisfies the condition that event (a) occurred before event (b).

An event record combines the ***EventID,*** the ***Resource*** and the ***ResourceID*** to indicate that a business event has occurred on a system. The event record is part of a well-known resource ***Events*** that represents all events that have occurred on a system.

The change proposal adds to the existing Web API 1.1 specification:

-- Add a new section 2.6.1 –

2.6.1 Events

The Events resource is an endpoint that returns event records as defined below.

A producer MUST provide a Resource named **Events**

An Event Entity has three fields, the ***EventID***, the ***Resource*** name and the ***ResourceID*** that uniquely identifies an entity of **Resource** type.

The individual entities of the Events endpoint MUST have the form ***EventID***, ***Resource***, ***ResourceID***

The data type of ***EventID*** is the positive portion of ***int64***.

The data type of ***Resource*** is ***string***. It is the well-known name for the resource type.

The data type of ***ResourceID*** is the unique identifier for a specific record within **Resource** and thus must be a ***string***

The ***EventID*** MUST conform to the property that is is immutable and monotonic.

The ***Resource*** values in the entity MUST be described in the metadata.

The **Events** resource may contain all events that occur in a system or may contain events limited to those events or resources that a consumer has permission to view. The existence of an identified resource in **Events** does not change the visibility of the identified resource entity. That is controlled by the producer and the permission model they have implemented.

2.6.1.1 Workflow

Certain limitations are applied to the normal workflow of producing and consuming entities.

2.6.1.1.1 Producer

Producers MUST order **Events** in the time order that they occurred. ***EventID*** must have the property that it is unique, always increases in value and cannot change. Further, requesting the same ***EventID*** MUST result in the same ***Resource*** and ***ResourceID***. Wherever possible, Producers should order events to simplify referential integrity. For example, the addition of a **Property** Resource event should occur before any **Media** Resource events associated with the Property appearing in the **Events** Resource***.***

2.6.1.1.2 Consumer

Consumers use the current normal workflow to query

Example

A consumer wishes to get all the events after EventID 100

```
/Events?$filter=EventID gt 100
```

should expect a result in a form like this

```
[
  {EventID: 101, ResourceType: "Member", ResourceID: 21},
  {EventID: 103, ResourceType: "Property", ResourceID: 539},
  {EventID: 110, ResourceType: "Media", ResourceID: 1239},
]
```

There are two types of consumers and two cases comprise the workflow of a consumer. Most consumers will be synchronizing with the most current state of the system. A much smaller number of consumers will be collecting state changes throughout history to create analytics. The two cases of workflow are:

a. Initial synchronization.

The consumer is gathering events for the first time. Based on rules created by the producer or by the producer's client, the consumer

b. On going re-synchronization.

The consumer has previously been in synchronization and needs to 'catch up' with the current state of the system.

2.6.1.2 Handling Record Visibility

Based on business rules, Event records may change the availability and visibility for consumers based on the role of a consumer. For example, a consumer who had a role to receive IDX listings may not be permitted to only view listings that have a status of 'ACTIVE'. A consumer with this role would see an event that changes status and would then attempt to retrieve the record from the producer only to receive no record. This can be used by the consumer to know that the ResourceID they are attempting to retrieve is no longer part of the permitted records for their system and could take appropriate steps to remove that ResourceID from their visible record set.

Consumer Visibility Workflow

A consumer asks for the current set of **Events**. In this set is an event that changes a record that the consumer has from visible to not visible - we can think of this as a consumer delete event. The consumer does not know what has happened to the record, only that the record has changed state. When the consumer makes a normal query against the appropriate Resource, they will get a "No Record Found" response and can infer that the record and the child resources should be deleted. Based on section 2.6.1.1.1, the child records should have appeared at a lower ***EventID*** value than the parent record. This allows the consumer to remove references rather than deal with cascading deletes.

2.6.1.3 Relationship to HistoryTransactional

A separate change proposal adds the ***EventID*** to **HistoryTransactional** to associate events with entries in the **HistoryTransactional** resource.

Since **HistoryTransactional** is a history of business events for a system, a relationship exists between events in the **Events** resource and those that appear in **HistoryTransactional**. A consumer MUST, within the context of this standard, be able to interpret the relationship between **Events** and **HistoryTransactional**. Producers MAY, based on business rules, aggregate events such that the constraint should result in a one-to-one or a one-to-many relationship between **Events** records and **HistoryTransactional** resources.

Impact

Producers should expose a new resource that implements the requirements of the proposal.

Compatibility

As new functionality, this does not affect Compatibility

Certification Impact

Additional test rules will be needed.

001 - Check metadata for the existence of an Events resource of the form EventID, Resource, ResourceID. Additional fields in an Events resource will be ignored by the compliance test.

002 - Check metadata for the existence of an EventID field in the HistoryTransactional Resource.

003 - Confirm that data is returned for a GET on the Events endpoint.

004 - Confirm that for a sample set that any two records have sequentially increasing EventID values.

005 - Confirm that for a sample set of Events, that all records only contain Resource values that are defined in the metadata.

006 - Confirm that for a sample set of Events, values exist in HistoryTransactional.

007 - Confirm that the Events resource response is well formed.

008 - Confirm that for each record in a sample set of Events, that either: a record is returned OR a <<MISSING>> response value indicating that the record no longer exists OR a <<MISSING>> response value indicating that the record is not available to the consumer role is returned for each record.

009 - << Should we ensure sequence order and how would we describe this test? >>

010 - Confirm that requesting the same EventID results in the same combination of Resource and ResourceID.