EVE and API should indicate if an error includes an automatic retry

The EVE lifecycle management is based on eventual consistency. This means that if an state change to some object (device, app instance, etc) can not be performed then EVE will report the current operational state (which could indicating that something is in progress like a download) and also an error if there is some failure (such as a download failing, or the memory or adapter is currently used by some other app instance.)

However, in the EVE info API those are reported as errors; there is no indication that EVE might retry, and if so when, or whether it has already retried N times and it still fails.

This is a proposal for how to extend EVE and the EVE API to convey this information to the controller.

Examples of information EVE can provide

Download failed

Some error string (could be from DNS, HTTP, TLS, etc)

Will retry in N minutes (a timer which we can set with configItem)

Have retried M times already

Insufficient memory

Some error string about needed vs. available memory

Will retry when: some other app instance is halted and frees up memory

(Retry count might be less important here; each time an app instance is halted EVE will check if there is sufficient memory for this app instance but that isn't really a "retry" but a "check again" operation)

Missing hardware such as app-direct adapters

Some error string about app UUID XYZ using ethN (or USB)

Will retry when: app UUID XYZ is halted

(As above a retry count might not be useful.)

Examples where retry might not make sense

If the device model indicates that eth3 should exist we report an error. But since we don't support hot plug of hardware we are unlikely to ever retry. Hence such errors should not be retriable.

Also incorrect information (bad IP address string in some API which doesn't parse) would not be marked as retriable.

Possible API approaches

In the internal controller API we already have a severity field for the errors. This is never filled in from the EVE API since the EVE API does not have such a field.

One approach is to introduce a severity field in the EVE API (with values like ERROR, WARN, NOTICE) and use the NOTICE setting for things which will be retried. The EVE API would also have a retry condition (in X minutes, when resource Y is freed up) and perhaps also a retry count.

With such an approach we would need to add a retry-condition and retry-count to the internal controller API.

Furthermore, if retry-count reaches some large value (10?) or if the time since the original error exceeds some time (1 hour?), then maybe EVE or the controller should increase the severity from NOTICE to WARNING and later to ERROR. But if we do that we still need to report the retry_condition unless EVE at some point in time gives up. Current suggestion is to have EVE do this raising of the severity.

Implementation plan

Step0:

- Check if a download retry which fails updates the timestamp in the errinfo and whether that results in one event in the UI event log for each failed retry

Step1:

- Add severity enum to errinfo in EVE API - use NOTICE for things which will be retried. Need to define the values (NOTICE, WARNING, ERROR) is sufficient for now.
- Add retry_condition string to errinfo in EVE API
- Add object type enum to EVE API
    - We want to refer to app instances (which hold adapters), and potentially others
    - Should we define an enum with app instance, network instance, volume, content tree? (those are the key types in EVE)
    - Should we also define enum values for memory, adapter, inbound acl port conflict?
- Add referenced objects array to errinfo in EVE API; each entry has an object type and a UUID string.

Step2:

- Carry the above severity, retry_condition, and referenced_objects from the controller to the UI
- Make the controller event log include these with the severity field (bonus if the event log details has all if the info)
- UI does lookup of the type, UUID in the referenced_objects to display a name

Step3:

Look at whether we want a retry_count and other aspects from e.g, AWS Device Shadow service documents - AWS IoT Core in the EVE API. That is more related to EVE informing controller about pending changes and operationsthan the error/info reporting

Proto Changes:

To incorporate this change, the info.proto's errorinfo has been updated as follows:

```
message ErrorInfo {
  string description = 1;                   // error description
  google.protobuf.Timestamp timestamp = 2;  // Timestamp at which error had occurred
  Severity severity = 3;                    // Severity of the error
  repeated DeviceEntity entities = 4;       // list of objects referenced by the description
  string retry_condition = 5;               // condition to retry
}

Where Severity is a enum:

enum Severity {
  SEVERITY_UNSPECIFIED = 0; // severity unspecified
  SEVERITY_NOTICE = 1;      // severity notice
  SEVERITY_WARNING = 2;     // severity warning
  SEVERITY_ERROR = 3;       // severity error
}
```

and DeviceEntity is object of entityType and entityId:

```
message DeviceEntity {
  Entity entity = 1;           // entity type
  string entity_id = 2;        // entity uuid
}
```

Where Entity can be any of the following,

```
enum Entity {
  // Invalid Device Entity
  ENTITY_UNSPECIFIED = 0;
  // Base OS entity
  ENTITY_BASE_OS = 1;
  // System Adapter Entity
  ENTITY_SYSTEM_ADAPTER = 2;
  // Vault Entity
  ENTITY_VAULT = 3;
  // Attestation Entity
  ENTITY_ATTESTATION = 4;
  // App Instance Entity
  ENTITY_APP_INSTANCE = 5;
  // Port Entity
  ENTITY_PORT = 6;
  // Network Entity
  ENTITY_NETWORK = 7;
  // Network Instance Entity
  ENTITY_NETWORK_INSTANCE = 8;
  // ContentTree Entity
  ENTITY_CONTENT_TREE = 9;
  // Blob Entity
  ENTITY_CONTENT_BLOB = 10;
  // VOLUME Entity
  ENTITY_VOLUME = 11;
}
```

Please note that entities like ENTITY_SYSTEM_ADAPTER, ENTITY_VAULT, ENTITY_ATTESTATION and ENTITY_PORT have their own entity ID even though, unlike the others, they do not have UUIDs.