Implementation Guidelines

Integration Error Handling & API Event Manager (IAM)

Release 1.0

Document Control

Version	Date	Author(s)	Description
1.0	**August 2023**	**Thomas Keene, Aaron Li, Dennis Nazarov**	MVP

Table of Contents

Introduction

The Integration Error Handling & API Event Manager application provides a solution for handling the responses of outbound integration calls and exposes an API for inbound event management.

Key Functionality:

- Inbound & outbound event auditing
- Integration error auditing, displaying and reporting
- API header and JSON body validation
- Inbound API event routing
- Outbound requests and asynchronous inbound response correlation
- Administrative features (data management, JSON Schema creation & upload)

For quick implementation reference guides, refer to the 'Calling Integrations - Steps' and 'Using the Event Manager API - Steps' sections.

Limitations

The application is not suitable for use cases where:

- It is not possible to isolate integration calls within process models to one integration per process. This limitation is imposed by the use of the process ID as the correlation ID. This limit could be alleviated by using alternative unique ID generation methods, but it is still recommended to isolate integration smart services in their own process models.
- Correlation ID must be returnable from external system to Appian via API header to access correlation functionality.
- JSON content validation has to be app specific (in subsequent processes).
- Event manager is only valid for asynchronous Appian subsequent processes (i.e. no data can be returned to the external system from the Appian API response).
- A single JSON schema can only be used for the body validation of a request event type (i.e., dynamic JSON schema validation is not supported).
- Only Content-Types of application/json are accepted for the Event Manager WebAPI.

Other limitations include:

- Event Manager Audit Synced Record: the event manager audit record is synced, and therefore is limited to 2,000,000 active rows (as of 23.2). IAM currently exposes functionality to manually deactivate event manager audit entries in bulk, however for large volume use cases, an automated, scheduled process to perform this activity is suggested to be developed.
- Database and System load testing performance have been documented separately and should be reviewed to ensure IAM is suitable for a use case before implementation. This

documentation should also be used to help developed revise any IAM process model data management policies as necessary.

Recommended Use Cases

Recommended use cases of the accelerator application include:

- Auditing of outbound calls to external systems through integration objects and/or inbound calls to Appian from external systems through an API object.
- Using integration objects, when it is important to know where an erroneous response has been received (either for the end-user who's action caused the error, or for some administrator)
- Exposing Appian to an external system through an API, where the inbound calls need to be routed according to the type of call (the event type)
- Exposing Appian to an external system where it is important to validate the JSON schema of the request body.
- Correlating outbound integration calls with asynchronous inbound API responses.

Installation

Prerequisites

- ☐ Appian 23.2 or later.
- ☐ Available plug-ins:
    - ☐ Appian Regular Expression Functions
    - ☐ JSON Tools
    - ☐ Content Tools
- ☐ Database:
    - ☐ Appian cloud MariaDB

IAM is designer such that it will not have circular dependencies with business applications, and should be deployed before business applications in the deployment order.

Some use cases for handling web APIs will require the Transaction Manager accelerator. Typically, the Transaction Manager comes into the picture when dealing with higher volumes, so that you can't safely start a process for every received message to a web API straight away. These requirements should be assessed and implemented separately as necessary. For advice on integrating IAM with TM, contact the document authors.

Application Import

1. Run the IAM DB script in the Appian Cloud intended database.
2. Import the IAM application, with the customisation file.
3. Verify & publish IAM Data Store, ensure all records are linked to the data source.
4. Add relevant users to the IAM Administrators user group.
5. Create a service account and add to the IAM API Service Accounts. If creating new / additional Web APIs, ensure security is set for the IAM API Service Accounts can
6. Review the IAM test results documentation, and use this to revise the IAM process model data management policies, as necessary.
7. Use the IAM Testing Documentation to revise the data management policies if IAM process models, according to expected volumes.

When deploying up environments, the database script should be run each time if there are changes to the reference data (IAM_PROCESS_MODEL or IAM_REF_INTEGRATION_ERROR_TYPE tables).

Group Configuration

Group Name	Access
IAM Administrators	- Access to the Integration and API Manager Site. - Admin access to objects in the environment. - Admin to the ‘IAM Inbound Event Manager’ API object.
IAM Users	- Viewer access to objects in the environment.
IAM API Service Accounts	- Viewer to the ‘IAM Inbound Event Manager’ API object.

User Documentation

Sub-Process Security

Note that all re-usable sub-processes are configured to run with IAM All Users as a viewer group. Therefore, users of business applications should also be added to IAM All Users, whilst avoiding circular dependencies (adding individual users, or adding a member group that sits in an application before IAM in the deployment order).

Configuring Reference Data & Constants

The following section lists out all the objects within the IAM application that should be changed to configure the IAM application. Subsequent sections may give more guidance as to when updates to these objects may be necessary:

- DB Tables, populated using the idempotent SQL scripts to add rows to the following tables as needed:
    - **IAM_PROCESS_MODEL**; stores the list of downstream business processes and their UUIDs, so that inbound API calls can be routed as needed.
    - **IAM_REF_INTEGRATION_ERROR_TYPE**; allows storage of an error description and error type, that can be used to return standard error handling messages to end users based on the event type, or by custom logic.
- Constants:
    - **IAM_TEXT_LIST_VALID_SOURCE_SYSTEMS**; stores the list of accepted values for the source system API header.
    - **IAM_TEXT_ENTITY_TYPE_DESCRIPTIONS**; stores user-friendly, readable string values for the entity types that may be stored in the ENTITY_TYPE column of the IAM_INTEGRATION_ERROR table.
    - **IAM_INT_MAX_ERRORS_WEEKLY**; constant holding the maximum expected number of weekly integration errors, to keep interfaces performant. 1000 on app import.
    - **IAM_INT_MAX_REPORT_GROUPINGS**; constant holding the maximum number of groups for integration error chart reports (event types & ref integration error types). 25 on app import.
- Expression Rules:
    - **IAM_listEventTypes**; expression rule to hold a list of constants of accepted event types, for the API header validation (should be unique values).
    - **IAM_listEntityTypes**; expression rule to hold a list of constants of entity types to be stored in the ENTITY_TYPE column of the IAM_INTEGRATION_ERROR table (should be unique values).
    - **IAM_formatEntityId**; an expression rule to format a user-friendly layout string for related records, used to display which record an integration error relates to.
- Decision Tables:
    - **IAM_DT_routeInboundEventManagerCalls**; to map inbound event types to the subsequent business process to launch.
    - **IAM_DT_returnFunctionalityStatus**; to determine which pieces of functionality are switched on / off within the IAM app.
    - **IAM_DT_retutrnJsonSchemaFromEventType**; to map inbound event types to JSON schemas from which to validate the request body.
    - **IAM_DT_returnIntegrationErrorTypeFromEvent**; to map outbound event types to the relevant reference integration error type ID if an integration error occurs when an outbound call to an integration object is made.

- **IAM_DT_mapEntityTypeToDescription**; to map the unique entity type labels to their readable, user-friendly strings.

Calling Integrations - Steps

When calling an integration smart service within a business process, the following steps are recommended:

1) As a custom output on the integration smart service node, populate:
   ```
   rule!IAM_populateCommonIntegrationErrorStandard(
   isSuccess: ac!Success,
   appianIntegrationError: ac!Error,
   service:""SERVICE_CONSTANT"",
   correlationId: pp!id  /*(limit of 1 integration per process / subprocess). */
   httpResponseCode: ac!Result.statusCode
   )
   ```
   Replacing ""SERVICE_CONSTANT"" with a newly created constant in the IAM app, to indicate the service used by the integration, named IAM_TEXT_SERVICE_<name>.

   Save into a process variable, e.g. integrationError, of type:
   IAM_U_COMMON_INTEGRATION_ERROR_DETAILS

   Note that this expression rule extracts the error title, summary and details returned by the integration object in the default error handling response. If more specific details are required, the 'Override and define all error conditions' error handling setting should be configured within the integration object itself.

   Alternatively, for the correlationId, it may be a requirement for external systems to use UUIDs, and therefore inputs of type text are also accepted if an alternative method of establishing a unique identifier is used.

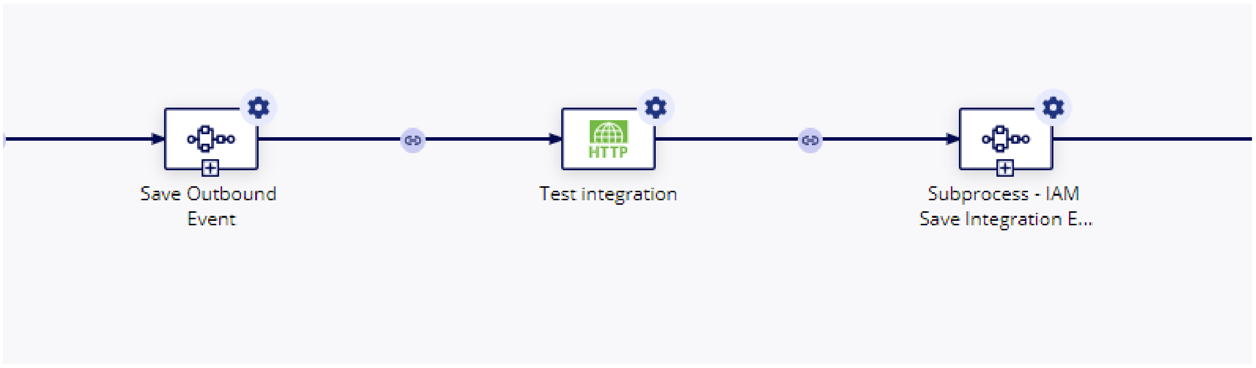
2) "Add a 'Save Outbound Event Manager Audit' sub-process node before the integration object, run synchronously and return the event ID into a new process variable. Configure parameters: *for more details on process variables, see Processes & Variables section.
   - 'correlationId' is the process ID (pp!id)
   - 'actionedBy' is a user (typically pp!initiator)
   - 'eventType' is a constant created within the IAM app, that represents the type of event that is occurring (e.g. 'CREATE_CASE').
   As an output from the process mode:
   - eventManagerAuditId (required in step 3)

3) Add a 'Save Integration Error' sub-process node directly after the integration object, regardless of if flow should continue if an error occurs (handle this separately after the sub-process node). Configure parameters:
   - 'createdBy' as a user (pp!initiator)
   - 'entityId' is a linked record primary key ID, if one is applicable.
   - 'eventManagerAuditId' as the output from step 2.
   - 'eventType' (as per step 2)
   - 'integrationErrorUtilityCdt' is a variable of variable of type IAM_U_COMMON_INTEGRATION_ERROR_DETAILS, which is the output of point 1.
   '- isDismissed' is a boolean, to determine if any error recorded should be dismissed by default or not.
   - 'overrideErrorTypeId' if the desired reference integration error type ID should be determined by custom logic. Otherwise, the output from the decision table IAM_DT_returnIntegrationErrorTypeFromEvent will be used.
   - 'showSyncInterface' is a boolean, set to true if you choose to show a user a chained interface, to communicate if a synchronous error occurs.
   - 'relatedEntity' is a linked record type, created as a new, unique constant in the IAM application if one is applicable.

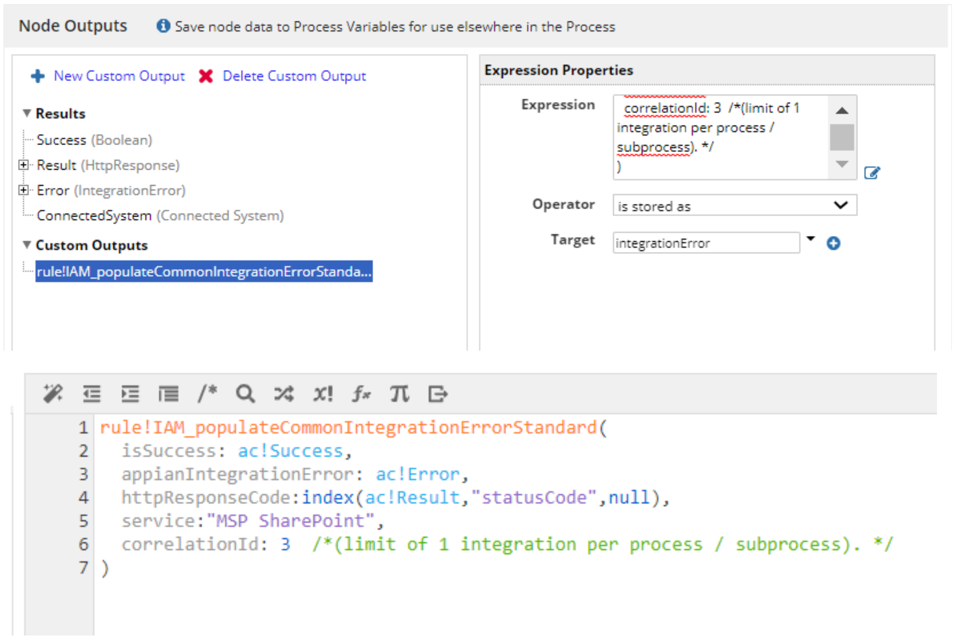
4) Add the event type constant (if new) to IAM_listEventTypes.

5) **[Optional]** Map the event type to the error type, if the same event type should always return the same integration error type, in IAM_DT_retutrnIntegrationErrorTypeFromEvent.
6) **[Optional]** Add the **relatedEntity** constant (if new) to IAM_listEntityTypes, add a description to IAM_TEXT_ENTITY_TYPE_DESCRIPTIONS, and map the description in IAM_DT_mapEntityTypeToDescription
7) **[Optional]** Add related entity ID formatting logic to IAM_formatEntityId, if displaying errors in interfaces.
8) **[Optional]** Where desired, add reference integration error type data to the IAM_REF_INTEGRATION_ERROR_TYPE table. Either pass directly in the 'Save Integration Error' sub-process node inputs, or configure the error type to be determined from the event type in the decision table IAM_DT_returnIntegrationErrorTypeFromEvent.

Example implementation:



Sub-Process Nodes Configuration



Results
- Success (Boolean)
- Result (HttpResponse)
- Error (IntegrationError)
- ConnectedSystem (Connected System)

Custom Outputs
- rule!IAM_populateCommonIntegrationErrorStanda...

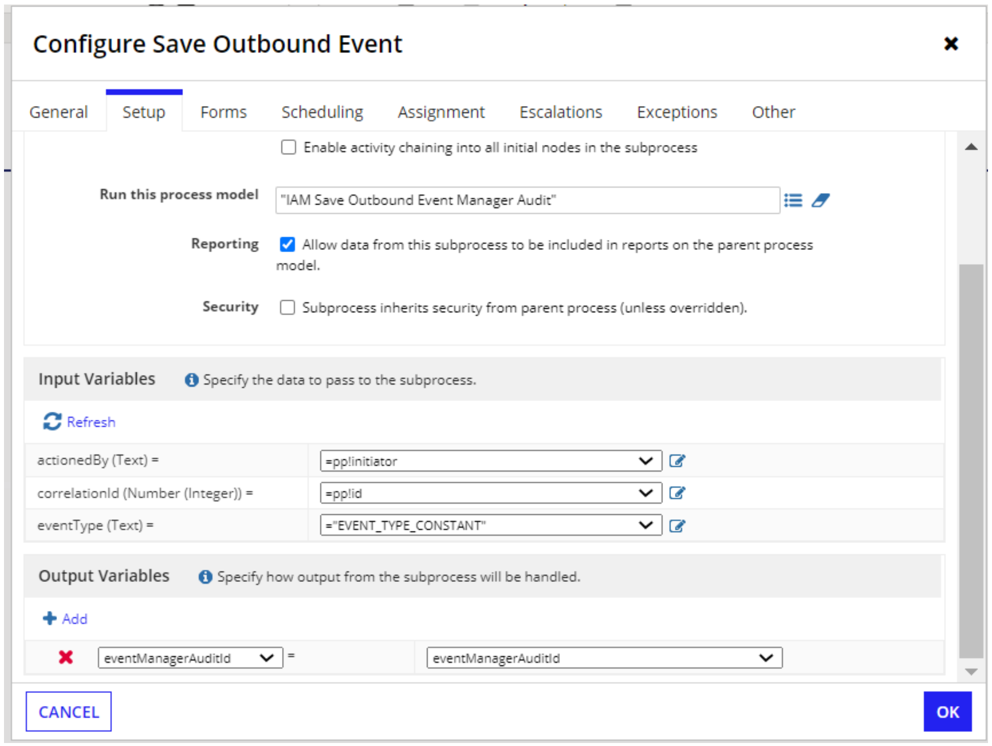
Expression: correlationId: 3 /*(limit of 1 integration per process / subprocess). */)

Operator: is stored as

Target: integrationError

```
rule!IAM_populateCommonIntegrationErrorStandard(
  isSuccess: ac!Success,
  appianIntegrationError: ac!Error,
  httpResponseCode:index(ac!Result,"statusCode",null),
  service:"MSP SharePoint",
  correlationId: 3  /*(limit of 1 integration per process / subprocess). */
)
```

Integration Smart Service Node Custom Output



Configure Save Outbound Event

General | Setup | Forms | Scheduling | Assignment | Escalations | Exceptions | Other

☐ Enable activity chaining into all initial nodes in the subprocess

Run this process model "IAM Save Outbound Event Manager Audit"

Reporting ☑ Allow data from this subprocess to be included in reports on the parent process model.

Security ☐ Subprocess inherits security from parent process (unless overridden).

Input Variables — Specify the data to pass to the subprocess.

Refresh

actionedBy (Text) =	=pp!initiator
correlationId (Number (Integer)) =	=pp!id
eventType (Text) =	="EVENT_TYPE_CONSTANT"

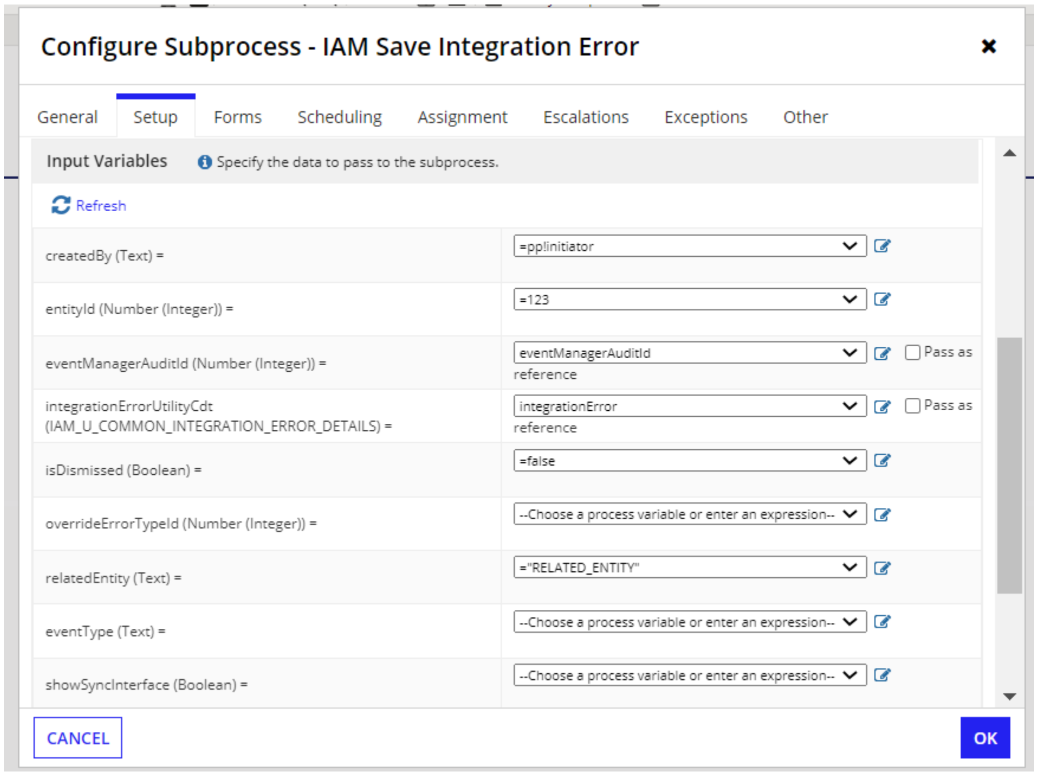
Output Variables — Specify how output from the subprocess will be handled.

Add

eventManagerAuditId =	eventManagerAuditId

CANCEL | OK

Save Outbound Event Sub-Process Node Configuration



Configure Subprocess - IAM Save Integration Error

General | Setup | Forms | Scheduling | Assignment | Escalations | Exceptions | Other

Input Variables — Specify the data to pass to the subprocess.

Refresh

createdBy (Text) =	=pp!initiator
entityId (Number (Integer)) =	=123
eventManagerAuditId (Number (Integer)) =	eventManagerAuditId ☐ Pass as reference
integrationErrorUtilityCdt (IAM_U_COMMON_INTEGRATION_ERROR_DETAILS) =	integrationError ☐ Pass as reference
isDismissed (Boolean) =	=false
overrideErrorTypeId (Number (Integer)) =	--Choose a process variable or enter an expression--
relatedEntity (Text) =	="RELATED_ENTITY"
eventType (Text) =	--Choose a process variable or enter an expression--
showSyncInterface (Boolean) =	--Choose a process variable or enter an expression--

CANCEL | OK

Save Integration Error Sub-Process Node Input Configuration

Using the Event Manager API - Steps

Note that for more complex environments, the provided IAM Event Manager Web API may not satisfy all use cases. Similar API objects can be created by developers as required, using the Event Manager API as a template for how IAM functionality can be utilised.

When anticipating calls from external systems to Appian via the event manager API, the following steps are suggested to configure:

1) Add the expected API header value to IAM_TEXT_LIST_VALID_SOURCE_SYSTEMS for source systems, to validate successfully when inbound API requests are made.
2) Create a constant with the expected API header value for the event type, and add the constant to the expression IAM_listEventTypes.
3) If the inbound call is required to be routed to a downstream business process:
    a) Add the business process to the IAM_PROCESS_MODEL DB table, using the idempotent scripts.
    b) Configure the routing in IAM_DT_routeInboundEventManagerCalls
4) If the inbound call request body should be validated against a JSON schema:
    a) Create a JSON schema using the Integration & API Manager Site, or alternatively upload an existing schema through the site upload feature.
    b) Use IAM_DT_retutrnJsonSchemaFromEventType to map the expected event type to the relevant JSON schema. Note that is no schema is mapped, the request body will just be checked that it is a valid JSON.
5) In downstream processes:
    a) Use IAM_QE_getEventManagerAudit to query data from the saved inbound event, including the payload, to extract relevant data downstream.
    b) If relevant, integration errors can be configured downstream too, by first creating an ER to populate the IAM_U_COMMON_INTEGRATION_ERROR_DETAILS CDT from the inbound payload, and then following the Calling Integrations - Steps section from step 3.

Integration & API Manager Site

For users in the IAM Administrators group, a site is available to monitor integration errors and perform administrative functions:

Dashboard

The dashboard provides a high-level, weekly overview of integration errors, and offers a chance to explore the data to highlight where there may be problematic services or events.

Integration Errors

The integrations error page is a record list of all integration errors, from which errors can be bulk dismissed.

Configurator

There are two sections to the configuration page:

- JSON Schemas: pre-made schemas can be uploaded directly, or basic schemas (with up to 1 level of nesting) can be created directly from an intuitive JSON schema constructor.
- Data management: if the number of event manager audit entries approaches 2,000,000 (synced record limits), this page can be used to manually deactivate entries in the table, according to the creation data of the events. However, in high-volume environments, a custom scheduled process to deactivate events is recommended for scalability.

Appendix

API Response Codes

400	The event type is not known	{ "success": false, "details": [{ "error": "INVALID_EVENT_TYPE", "message": "Unknown event type." } }
400	The source system is not known	{ "success": false, "details": [{ "error": "INVALID_SOURCE_SYSTEM", "message": "Unknown source system." } }
	Source system timestamp invalid format	{ "success": false, "details": [{ "error": "INVALID_SOURCE_SYSTEM_TIMESTAMP_FORMAT", "message": "The Source-System-Timestamp header is not of a valid format, e.g. 2012-04-23T18:25:43.511Z" } }
	Invalid request body	{ "success": false, "details": [{ "error": "INVALID_REQUEST_BODY", "message": "The request body is an invalid JSON." } }
400	Multiple errors	{ "success": false,

		"details": [
		{
		"error": "INVALID_EVENT_TYPE",
		"message": "Unknown event type."
		},
		{
		"error": "INVALID_SOURCE_SYSTEM",
		"message": "Unknown source system."
		}
		]
		}

401	Authentication failed	(Appian native)
403	Access forbidden	(Appian native)
404	There is no Web API with the specified endpoint and HTTP method	(Appian native)
404	The user is not in the viewer role or higher for the Web API	(Appian native)
500	There was an error evaluating the Web API's expression	(Appian native)
500	The result of the expression evaluation was not an HTTP Response object	(Appian native)

200	Generic success response	{
		"success": true
		}

Database Tables

IAM_EVENT_MANAGER_AUDIT

Field Name	Field Type	Description
EVENT_MANAGER_AUDIT_ID	int(11)	Primary Key
REQUEST_TYPE	varchar(255)	Used to identify either INBOUND (calls to Appian APIs) or OUTBOUND (calls from Appian through an integration object) events
ACTIONED_BY	varchar(255)	A field that identifies an individual user account who has actioned on the event manager audit record.

		For inbound request types, if the event can be correlated, this value will be populated as the same value of the corresponding outbound event.
CORRELATION_ID	varchar(255)	An alpha-numeric field, used to correlate outbound and inbound calls. Outbound calls should be configured to used Appian process ID's, pp!id.
SOURCE_SYSTEM	varchar(255)	[INBOUND request types only] Used to indicate the system from which the call to the Appian API is made.
SOURCE_SYSTEM_TIMESTAMP	datetime	[INBOUND request types only] The timestamp from the source system from calling the Appian API.
EVENT_TYPE	varchar(255)	A custom text used to identify the event type by the appian designer.
IS_SUCCESS	boolean	[INBOUND request types only] Identifies if inbound requests were successfully validate by the API logic.
ERROR_SUMMARY	JSON	[INBOUND request types only] If IS_SUCCESS is false, stores the Appian API response details.
PAYLOAD	JSON	[INBOUND request types only] The request body sent in the request to the Appian API.
CREATED_ON	datetime	A field to capture when the record was created.

IAM_INTEGRATION_ERROR

Field Name	Field Type	Description
INTEGRATION_ERROR_ID	int(11)	Primary Key
INTEGRATION_ERROR_TYPE_ID	int(11)	FK reference to the IAM_INTEGRATION_ERROR_TYPE table.
EVENT_MANAGER_AUDIT_ID	int(11)	FK reference to the EVENT_MANAGER_AUDIT table.
RELATED_ENTITY_ID	int(11)	For storing the PK of related records, for which the error applies.
SERVICE_NAME	varchar(255)	Represents the API service that failed.
IS_DISMISSED	boolean	Indicates if a user has dismissed the error, or not.
CREATED_ON	datetime	A field to capture when the record was created.
CREATED_BY	varchar(255)	A field to capture who created the record.
UPDATED_ON	datetime	A field to capture when the record was updated.
UPDATED_BY	varchar(255)	A field to capture who updated the record.
ERROR_DETAIL	varchar(255)	The detail attribute returned by an integration object, when the integration call is unsuccessful, containing details of the failure.
ERROR_MESSAGE	varchar(255)	The message attribute returned by an integration object, when the integration call is unsuccessful, containing details of the failure.
ERROR_TITLE	varchar(255)	The title attribute returned by an integration object, when the

		integration call is unsuccessful, containing details of the failure.
HTTP_RESPONSE_CODE	int(11)	The http response code returned by the API

IAM_PROCESS_MODEL

Field Name	Field Type	Description
PROCESS_MODEL_ID	int(11)	Primary Key.
UUID	varchar(255)	The UUID of the process model appian object.
CREATED_ON	datetime	A field to capture when the record was created.
CREATED_BY	varchar(255)	A field to capture who created the record.
UPDATED_ON	datetime	A field to capture when the record was updated.
UPDATED_BY	varchar(255)	A field to capture who updated the record.
IS_ACTIVE	boolean	A field determines whether the process model is active and can be reference in IAM.
LABEL	varchar(255)	A fied that is used to name the process model(s).

IAM_INTEGRATION_ERROR_TYPE

Field Name	Field Type	Description
INTEGRATION_ERROR_TYPE_ID	int(11)	Primary Key.
LABEL	varchar(255)	A fied that is used to label the specific integration error types that may exist.
ERROR_DESCRIPTION	varchar(255)	A field that is used to provide a description on the integration error, to give end users more context.
CORRECTIVE_ACTION	varchar(255)	The suggested action to resolve the integration error that can be used to display to end users.
CREATED_ON	datetime	A field to capture when the record was created.
CREATED_BY	varchar(255)	A field to capture who created the record.
UPDATED_ON	datetime	A field to capture when the record was updated.
UPDATED_BY	varchar(255)	A field to capture who updated the record.
IS_ACTIVE	boolean	A field determines whether the integration error type is available or unavailable for use.

IAM_RELATED_ENTITY

Field Name	Field Type	Description

RELATED_ENTITY_ID	int(11)	Primary Key - text to also accept unique identifiers o
ENTITY_TYPE	varchar(255)	A string that reprensents the type of entity, for which the entity ID identifies.
ENTITY_ID	varchar(255)	The unique identifier of the related entity
CREATED_ON	datetime	A field to capture when the record was created.
CREATED_BY	varchar(255)	A field to capture who created the record.
UPDATED_ON	datetime	A field to capture when the record was updated.
UPDATED_BY	varchar(255)	A field to capture who updated the record.
IS_ACTIVE	boolean	A field determines whether the integration error type is available or unavailable for use.

Processes & Variables

Save Outbound Event Manager Audit

Process Parameters:

- actionedBy; the username to record as actioning the event. Typically, use pp!initiator.
- correlationId; a unique correlation ID for correlation any subsequent inbound events to the initiating outbound event. Typically, use pp!id.
- eventType; a constant to refer to the outbound event type, expected to be passed as a constant. An example would be 'CREATE_CASE'.

Anticipated Process Outputs:

- eventManagerAuditId; to be saved into an integer process variable, to be passed to the Save Integration Error node.

Save Integration Error

Process Parameters:

- createdBy; the string to record in the createdBy attribute of the record.
- entityId; [optional] the primary key ID of the entity which you wish to relate the error to.
- relatedEntity; [optional] a constant to represent the type of entity which relates to the error. An example would be 'CASE_RECORD'.
- eventType; a constant to refer to the outbound event type, expected to be passed as a constant. Expected to be the same as the value passed to the earlier 'Save Outbound Event Manager Audit' sub process. An example would be 'CREATE_CASE'.
- eventManagerAuditId; the event manager audit ID for the event from which the integration error occurred. Expected output value passed from the earlier 'Save Outbound Event Manager Audit' sub process.
- integrationErrorUtilityCdt; a utility CDT containing data on the success or failure of the integration smart service.

- isDismissed; a boolean that can determine if the integration error entry should be dismissed on creation or now.
- overrideErrorTypeId; [optional] an integer pointing to the reference integration error type table, if custom logic is used to determine this ID. Otherwise, the error type ID is returned from the event type in IAM_DT_returnIntegrationErrorTypeFromEvent.
- showSyncInterface; a boolean that can determine if a chained interface showing summary details of the error should be shown or not.

Anticipated Process Outputs:

- None.