Appian Case Management Studio

Common Customization and Configuration Guidance

Table of Contents

Notice of Rights

Overview

There are customizations and extensions we know you will want to make to Appian Case Management Studio (CMS) in order to tailor it for a given use case. Whether this is in support of a prospective demo, a customer implementation, or your own efforts to become familiar with CMS, this guide will point you in the right direction.

We've documented high level steps for each of these changes in an effort to streamline this process and make it as formulaic as possible. Changes are listed in the order we recommend you make them in. We suggest completing any necessary customizations in Appian designer before making configurations in the Studio module, since you may want to use some of the customized data in your configurations (e.g., using newly added data fields in Report Builder, or using newly added task types when creating workflows in Studio).

As always, we encourage you to leverage the Appian platform, your creative freedom and judgment to make further customizations and extensions beyond those outlined here.

Legend

The following syntax is used when referring to items in Appian Designer:

- Application Container names are bolded - e.g., **Case Management Studio Base Application**
- Appian Object names are underlined - e.g., CMGT_WorkspaceSite
- Database Table names are in uppercase - e.g., CMGT_CFG_ENTITY_TYPE

Common Customizations/Extensions

Add Custom Data Fields

Case Management Studio has several tables backing its base records:

- CMGT_CASE
- CMGT_TASK
- CMGT_ENTITY

Custom fields for CMGT_CASE can be added via the Studio site.

When adding custom fields for the other tables, consider if the data fields you are adding are common fields (i.e. applicable to all entity types in an entity category) or unique fields (i.e. entity type specific) to determine which table (e.g., entity category or entity type) they should be added to.

If the new fields are large in number and specific to particular entity or task type, then it's probably best to create a new table with a relationship to the base record:

1. Create a new record type with new fields
2. On the new record type add relationship to base record, name the new field the same name as the base record primary key
3. Add a foreign key (FK) to the base record type (perform this on the db table)
4. Save changes and confirm new record type is created as expected
5. Go to base record type and add relationship to new record type
6. Save Changes

Now the new fields will be accessible via the relationship to the base record. If the relationship is 1:1 or 1:M, then any writes will work on the new record as expected. For any other relationship types, process changes may be necessary to write the new record appropriately.

See each specific section (e.g., entity or task) to learn more about what specific rules will need to be modified when adding data for each.

Leverage Email Templates

By default, the CMS ships with several consumer-grade email templates which are used for several out of the box notifications (e.g., sending confirmation of case creation). We encourage you to use these templates as a starting point for any additional customizations involving email notifications. The templates give you a head start with branding and formatting information for display in emails. You may also decide to update the email format. How to accomplish these customizations is outlined below.

Update Email Format

CMGT_EmailTemplate_Base is the template used for the general layout of all emails. This document must be updated in order to reflect any layout updates. Branding updates are reflected from the Solution Hub.

New Email Notifications

New email notifications require two rules. One rule for general email settings such as recipients, subject etc, and another for updating the body of the base template. For examples of each rule, see CMGT_Portal_Email_EmailSettingMap_CreateCase and CMGT_Portal_Email_CaseCreation_ReplacementParamsMap respectively from the CMGT Portal Add-On.

In order to add an additional notification, follow the steps below:

1. Create a rule called CMGT_Email_XXX_ReplacementParamsMap with parameters for the genBundle and the required data needed for the body
2. Add a local variable for the applicable branding map
3. Create an array with the following:

```
Unset
{
/*This creates the find replace dictionary used to update the base template values
with any information specific to the new notification*/
  rule!CMGT_Email_BaseTemplate_ReplacementParamsMap(
   brandingMap: local!brandingMap,
   genBundle: ri!genBundle,
   instructionSize: "14px",
   instruction: **NEW_VALUES_HERE**,
   instructionColor: local!brandingMap.SecondaryTextColor,
   footerText: **NEW_VALUES_HERE**,
   buttonLabel: **NEW_VALUES_HERE**
   buttonLinkUrl: **NEW_VALUES_HERE**
  ),
/*Body*/
/*This section replaces the body of the email with html specific to the new
notification. If there's is a particular html template being used for formatting, the
replacement can be done using rule!CMGT_UTIL_FillDocumentTemplate*/
 {
  find: "body",
  replace:{
    **NEW_VALUES_HERE**
}

 }
}
```

4. Save rule
5. Create a new rule CMGT_Email_EmailSettingMap_XXX with rule inputs for relevant variables

6. Ensure the new rule has the following map structure as an output:

```
Unset
a!map(
 recipients: {**NEW_VALUES_HERE** },
 subject: **NEW_VALUE_HERE**,
 /*The recipientEmailAddress field is only necessary if the recipient is not a user or
a group*/
 recipientEmailAddress: **NEW_VALUES_HERE**,
 replacementParams: {
       /*Insert rule created in step 1*/
}
)
```

7. Identify the process model where the notification will be triggered
8. Add a subprocess for CMGT_SUB_SendEmail
9. Pass in rule created in step 5 to the sub process
10. Save & Publish
11. Test to ensure notification is working as expected

Adjust Due Date Logic

In CMS, due dates are assigned to both cases and tasks. Due dates are calculated using the SLA business days upon case or task creation. If no SLA is specified, then the due date will be null until manually set by a user. Due dates are stored in the DB as a GMT timestamp which represents the Due Date at 11:59:59 in the system timezone in order to be inclusive of the existing day. When the due date is displayed on the front end, the due date is displayed in the system timezone.

We manage calculating, displaying, and converting all datetime fields with the following rules:

- CMGT_UTIL_CalculateDueDateBasedOnDuration
- CMGT_UTIL_AddBusinessDays
- CMGT_UTIL_FormatDateTime

Add Entity Categories & Types

Entities are the "nouns" of your case management use case. They conceptually describe a person, business, place, thing, or concept that is relevant to a case. Entities have their own entity record, and each type and category has its own record to supplement the base entity record

By default, Case Management Studio ships with a default entity category of "People" and a default entity type of "Contact".

For an investigative use case, you may have entity categories such as people, agencies, or legal services. You may have entity types such as subjects, witnesses, and victims ("people" category) and police departments and detective units ("agencies" category).

This guidance goes over how to extend CMS to add additional entity categories and entity types.

E.g., I want to add Permits and Parcels as their own entity categories and entity types so that I can relate them to other entities (e.g., contractors, permits, and parcels) and cases. I also want to be able to display specific information about these entity types on the associated entity record list and entity summary page.

Create a new Entity Category

1. Record Type Creation and Updates

First, we need to create a new record type and table for the new entity category. The goal of an entity category is to store common fields that will be used across entity types within this category. This entity category table does not contain any metadata, as metadata is stored in the

CMGT_Entity table. For the purposes of example, the below steps will be creating a “Business” entity category.

a. Create a new record type for the entity category
    i. Name - CMGT_[NameOfEntityCategory] (e.g., CMGT_BUSINESS)
    ii. Description - set as desired
    iii. Review Record Type Security
    iv. Configure Data Source - Select New Data Model
    v. Choose Data Source
    vi. Leave Data Sync Enabled
    vii. Add Fields
        1. [NameOfEntityCategory]Id (Number (Integer))(PK)
        2. entityId (Number (Integer))
        3. Add any fields that all entity types of this entity category will use
        4. Keep basic CRUD fields
    viii. Review
        1. Make sure that the checkbox “Create Table” is checked
        2. Save changes

We will add a relationship between this new record CMGT_[NameofEntityCategory] later after we add a unique constraint on the ENTITY_ID column in the database.

2. Database Setup

Instructions will be followed by example SQL scripts to reference.

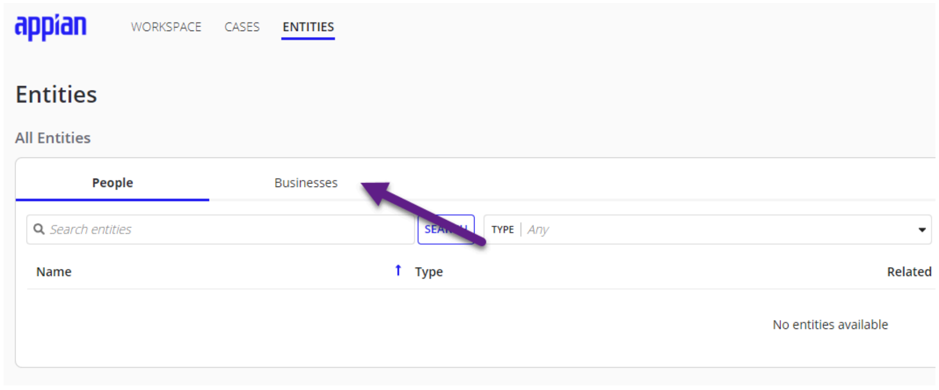
a. Insert a new row for the entity category
    i. This will allow all dropdown selections and configuration based logic to recognize the new Entity Category
    ii. Add a new row to CMGT_CFG_ENTITY_CATEGORY defining your Entity Category. Reference existing Entity categories to populate.

```
INSERT INTO `CMGT_CFG_ENTITY_CATEGORY`(
 `ENTITY_CATEGORY_ID`,
 `LABEL`,
 `LABEL_PLURAL`,
 `ICON`,
 `SORT_ORDER`,
 `IS_ACTIVE`,
 `CREATED_BY`,
 `CREATED_ON`,
 `MODIFIED_BY`,
 `MODIFIED_ON`
)
VALUES(
 1,
 'Person',
 'People',
 'user',
 1,
 b'1',
 CURRENT_USER,
 CURRENT_TIMESTAMP,
 NULL,
 NULL
);
```

   iii. Resync record type CMGT_CFG_EntityCategory

b. Make the entityId field unique
   i. Navigate to the db table corresponding to CMGT_[NameOfEntityCategory]
   ii. Click on the "Structure tab"
   iii. Select the row you want to make "unique" and click "Unique" at the bottom of the table
   iv. Resync record type CMGT_[NameOfEntityCategory]

c. At this point, you should see a new entity category within the Workspace UI:



3. Updates to "Write Category" objects

a. Modify CMGT_*Entity*
   i. Add a one to one relationship between CMGT_Entity and the newly created entity category record (e.g., CMGT_Business)
   ii. Name it "*[nameOfEntityCategory]*" (e.g., *business)*
   iii. Select a "One to One" relationship
   iv. Select "entityId" as the common record fields for the relationship
   v. Select "Write or delete CMGT_[nameOfEntityCategory] when modifying Entities"
   vi. Save the record type

b. Duplicate CMGT_Person_SetMetaData
   i. Rename it with the new entity category (e.g., CMGT_Business_SetMetaData)
   ii. Update the description
   iii. Change the rule input (name and type) to match the new entity category

        iv. For now you can comment out the map inside of the "childRelationships" parameter. This will be updated when creating a new entity type for this entity category.
   c. Modify CMGT_Entity_SetMetaData
        i. Under the "childRelationships" parameter, create a map for the new set metadata rule created above, following the CMGT_Entity.person example
   d. Modify CMGT_Entity_WriteRecordsProperties
        i. In the "fksToParent" rule input of the CMGT_UTIL_MergeWriteRecordIdsToProcessVariable rule, add the foreign key from your new Entity Category (e.g., CMGT_Entity.business.entityId)

4. Updates to "Query Category" objects

a. Modify CMGT_QR_GetEntity
    i. In the "fields" rule input of the CMGT_UTIL_QueryRecord rule, add the new category relationship CMGT_Entity.[categoryRelationshipName] (e.g., CMGT_Entity.business)
b. Create Entity Category Id Constant
    i. Duplicate CMGT_REFID_ENTITY_CATEGORY_PERSON to point to the Entity Category Id of your new Entity Category. Follow this naming convention.

Create a new Entity Type

1. Record Type Creation and Updates

Now that we've created a new entity category, we need to add at least one entity type to this category. The goal of an entity type is to store fields specific to that entity type. The entity type table does not contain any meta data, as that is taken care of in the entity table.

a. Create a new record type for the entity type
    i. Name CMGT_[NameOfEntityType] (e.g., CMGT_ArchitectureFirm)
    ii. Description - set as desired
    iii. Review Record Type Security
    iv. Configure Data Source (Select New Data Model)
    v. Choose Data Source
    vi. Leave Data Sync Enabled
    vii. Add Fields
        1. [EntityTypeName]Id (PK)
        2. Add any fields that are *specific* to this entity type
    viii. Add relationships
        1. Create a new relationship with the corresponding entity category table (e.g., CMGT_Business)

        2. Name it after the name of the category (e.g., “business”)
        3. Select a “One to One” relationship
        4. Name the field that will be added to the entity type table [EntityCategoryName]Id (e..g, “businessId”)
    ix. Review
        1. Make sure that the checkbox “Create Table” is Checked
        2. Save changes

b. Modify CMGT_[NameofEntityCategory]
    i. Add a new relationship
    ii. Select the CMGT_[NameOfEntityType] (e.g., CMGT__ArchitectureFirm) record
    iii. Name it after the name of the entity type (e.g., architectureFirm)
    iv. Select a “One to One” relationship
    v. Select “[entityCategoryId]” (e.g., “businessId”) as the common record fields for the relationship
    vi. Select “Write or delete CMGT_[nameOfEntityType] when modifying CMGT_[nameOfEntityCategory]"
    vii. Save the record type

2. Database Setup

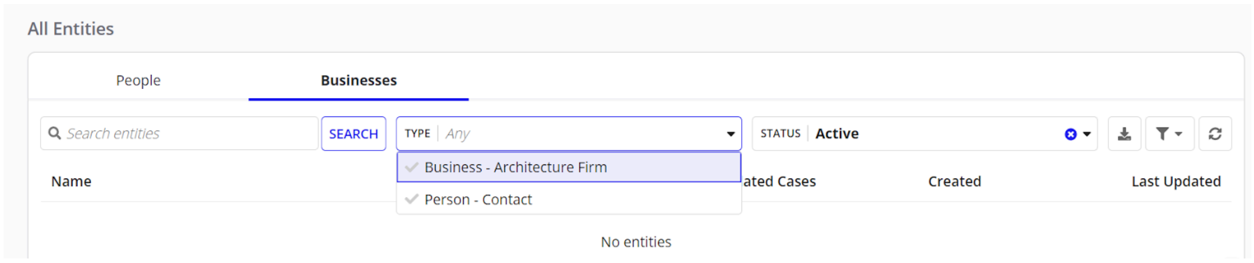
Instructions will be followed by example SQL scripts to reference.

a. Insert a new row for the entity type
    i. Add a new row to CMGT_CFG_ENTITY_TYPE defining your Entity Type. Reference existing Entity types to populate.

```
INSERT INTO `CMGT_CFG_ENTITY_TYPE`(
 `ENTITY_TYPE_ID`,
 `ENTITY_CATEGORY_ID`,
 `LABEL`,
 `LABEL_PLURAL`,
 `SORT_ORDER`,
 `ICON`,
 `IS_SYSTEM_ONLY`,
 `IS_ACTIVE`,
 `CREATED_BY`,
 `CREATED_ON`,
 `MODIFIED_BY`,
 `MODIFIED_ON`
)
VALUES (
 1,
 1,
 'Contact',
 'Contacts',
 1,
 NULL,
 b'0',
 b'1',
 CURRENT_USER,
 CURRENT_TIMESTAMP,
 NULL,
 NULL
);
```

b. Resync CMGT_CFG_ENTITYTYPE

c. At this point, you should see the new entity type in the type dropdown on the Workspace UI:

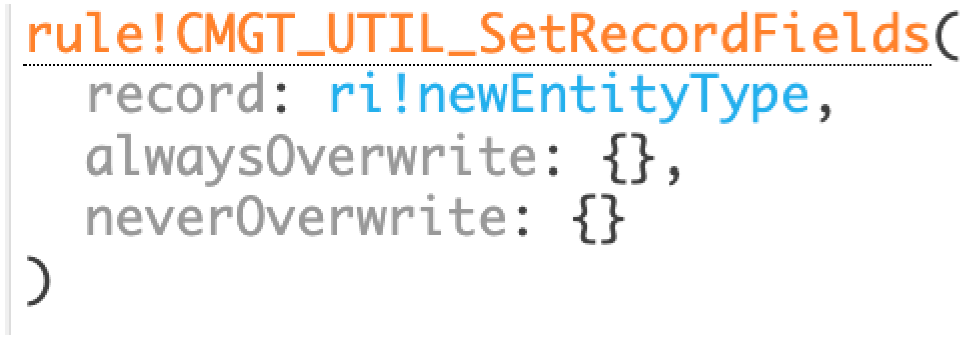


3. Collect entity type specific data to "Write Entity Type" objects

If there's data you need to collect about a specific entity type that cannot be covered in the entity category table then follow the steps below.

a. Create a new record type called CMGT_[EntityTypeCategory]_[EntityType] (ie CMGT_Person_Contact)
   i. Ensure there is a 1-1 relationship between the new record and entity category record

b. Create a rule called CMGT_[EntityTypeCategory]_[EntityType]_SetMetaData
   i. Add a rule input for the new entity type record
   ii. Call CMGT_UTIL_SetRecordFields and pass in the rule input as the record parameter



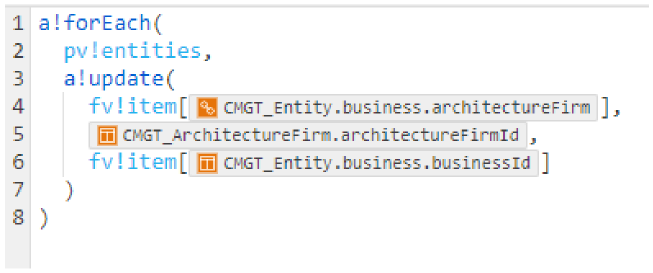
```
rule!CMGT_UTIL_SetRecordFields(
  record: ri!newEntityType,
  alwaysOverwrite: {},
  neverOverwrite: {}
)
```

c. Modify CMGT_[NameOfEntityCategory]_SetMetaData (e.g., CMGT_Business_SetMetaData)
   i. Under the "childRelationships" parameter, set the entity type meta data by passing in the following map

```
Unset
a!map(
 field:recordType!CMGT_[EntityCategory].relationships.[EntityTypeRelationship],
 value: rule!CMGT_[EntityTypeCategory]_[EntityType]_SetMetaData(
       newEntityType:ri!entityCategoryRecord.relationships.[EntityTypeRelationsh
ip]
       )
)
```

d. Create Entity Type Id Constant
   i. Duplicate CMGT_REFID_ENTITY_TYPE_CONTACT to point to the Entity Type Id of your new Entity Type . Follow this naming convention.

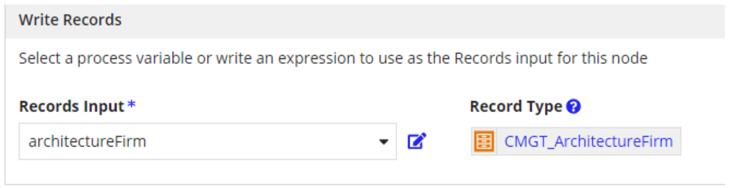
e. Create Entity Type Write Process
   i. Create a process model named CMGT_Entity_Create[EntityType]
   ii. Add a process variable parameter called entities of type CMGT_Entity list
   iii. Add a process variable for your new entity type that's of type multiple of the new entity type record
   iv. Add a script task to set the fk from the category record on the child entity type record. For example:



```
a!forEach(
  pv!entities,
  a!update(
    fv!item[CMGT_Entity.business.architectureFirm],
    CMGT_ArchitectureFirm.architectureFirmId,
    fv!item[CMGT_Entity.business.businessId]
  )
)
```

      1. Verify the Target is your Entity Type process variable

   v. Add a Write Record Type node to persist the new entity type
      1. In the Write node:
         a. Rename the Write node for your Entity Type
         b. For the "Records Input" field, your entity record should already be chosen in the dropdown, but you may have to select another value then change it back to get the appropriate Record Type to display



Write Records

Select a process variable or write an expression to use as the Records input for this node

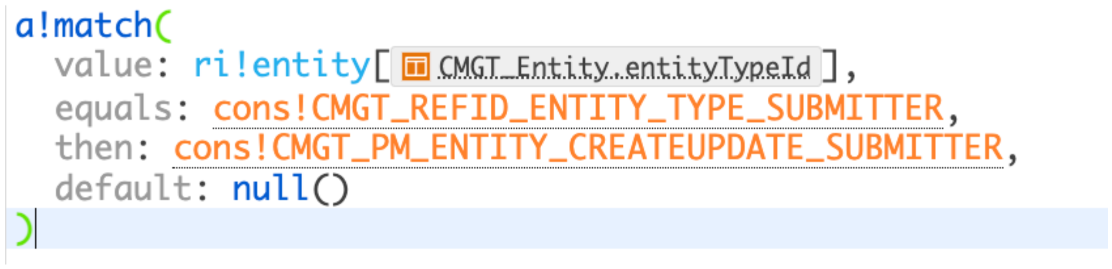
Records Input *	Record Type
architectureFirm	CMGT_ArchitectureFirm

   vi. Save and Publish your process model

f. Create a constant for the new entity type write process model called CMGT_PM_ENTITY_CREATEUPDATE_[ENTITYTYPE]. Follow this naming convention.
g. Modify CMGT_Entity_WriteRecordsProperties
    i. In the "relationships" rule input of the CMGT_UTIL_DropNonWritableRelationshipsFromRecord rule, add the relationship string to your new entity type by traversing the entity record and the entity category and type relationships
h. Modify CMGT_Entity_SetIdentifyingAttributeByEntityType
    i. In the match statement add a new "equals, then" parameters with the constant that points to the Entity Type Id and the field desired for the Identifying Attribute for that entity type. The identifying attribute is the field that can be used functionally to distinguish one entity type from another (for example, an identifying attribute for a person could be their email address if there are two people named "Joe Smith")
i. Modify CMGT_Entity_SetTitleByEntityType
    i. In the match statement add a new "equals, then" parameters with the constant that points to the Entity Type Id and the expression desired for setting the Title for that entity type
j. Create Entity Type Initialization Rule:
    i. Duplicate CMGT_UTIL_InitializeContactEntity for the new entity type
    ii. Alter the fields to be the appropriate Entity Category and Type Records
    iii. Change Entity Type Id to use the constant that points to the Entity Type Id
k. Modify CMGT_UTIL_InitializeEntityType
    i. In the match statement add a new "equals, then" parameters with the constant that points to the Entity Type Id and the new expression rule

NOTE: The following only needs to be completed the first time an entity type is set up

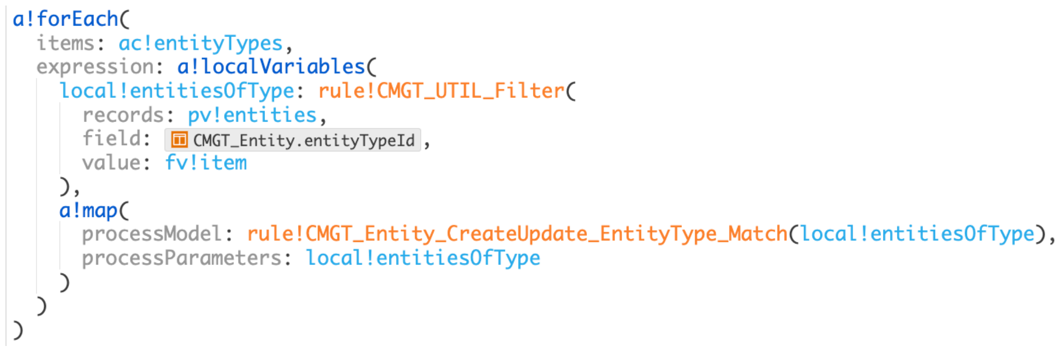
l. Create expression rule called CMGT_Entity_CreateUpdate_EntityType_Match
    i. Execute a match statement where the value is the entity record entity type id. Based on the entity type id then return the appropriate process model



```
a!match(
  value: ri!entity[CMGT_Entity.entityTypeId],
  equals: cons!CMGT_REFID_ENTITY_TYPE_SUBMITTER,
  then: cons!CMGT_PM_ENTITY_CREATEUPDATE_SUBMITTER,
  default: null()
)
```

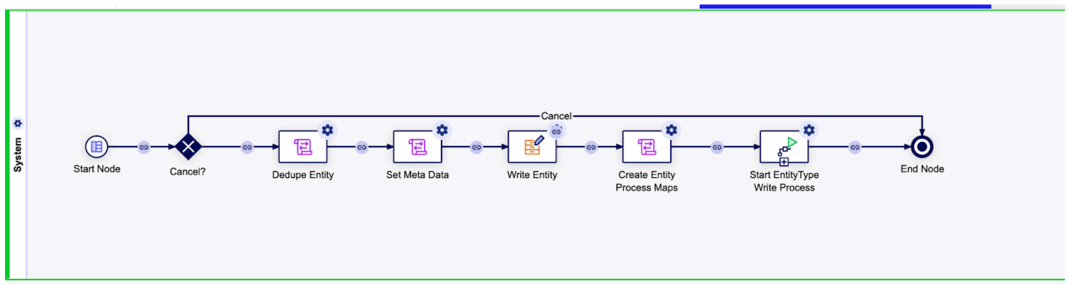
m. Update CMGT_Entity_CreateUpdate process model
    i. Add a script task that evaluates CMGT_Entity_CreateUpdate_EntityType_Match and returns the appropriate process parameters to be saved into a map process

variable



```
a!forEach(
  items: ac!entityTypes,
  expression: a!localVariables(
    local!entitiesOfType: rule!CMGT_UTIL_Filter(
      records: pv!entities,
      field: CMGT_Entity.entityTypeId ,
      value: fv!item
    ),
    a!map(
      processModel: rule!CMGT_Entity_CreateUpdate_EntityType_Match(local!entitiesOfType),
      processParameters: local!entitiesOfType
    )
  )
)
```

ii. Add a start process node to execute the appropriate process models based on the output of the script task

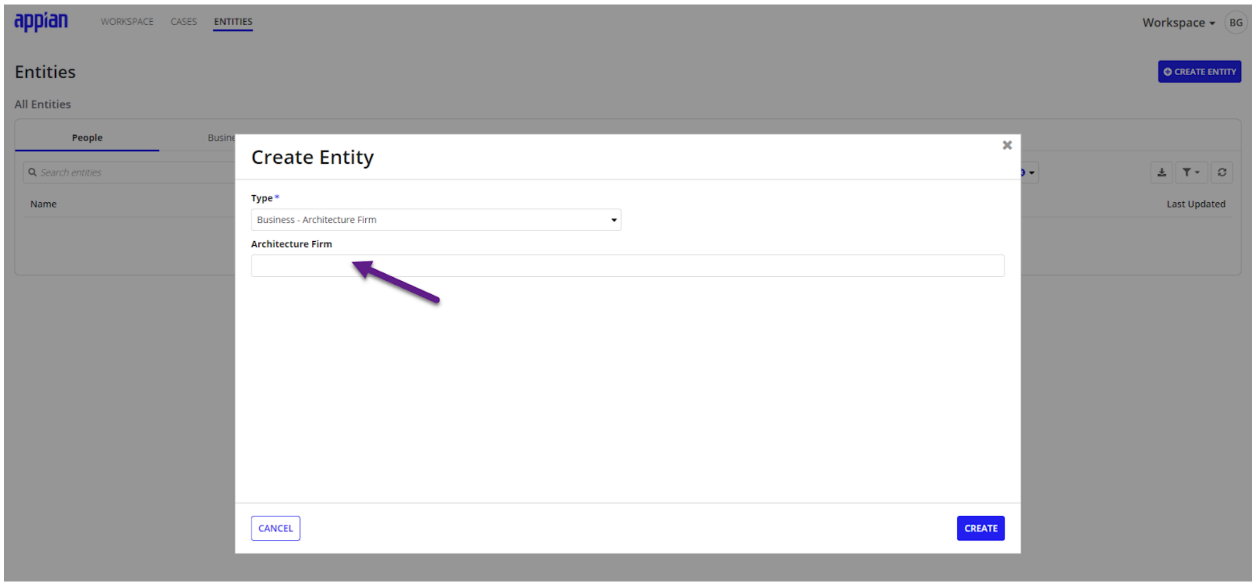


4. Updates to "Describe Entity" or "Entity Intake" objects

a. Create a new interface to input Entity Type Details
   i. Name: CMGT_[NameOfEntityType]_Details
   ii. Example of Incorporation and Fields: CMGT_Contact_Details
   iii. Make sure that it takes in the rule input of entity to match CMGT_Entity_RecordAction_CreateUpdate where it will be displayed

b. CMGT_Entity_Details
   i. In the match statement add a new “equals, then” parameters with the constant that points to the Entity Type Id and the new interface to display

c. Confirm that the new fields show up when the entity type is selected on the "Create Entity" dialog:



d. Confirm that submitting a new entity entry adds a row to the entity record list

5. Updates to "Display Entity" objects

a. If it's desired to display details of this new entity type in the case record summary view, then complete the following steps:
    i. Duplicate CMGT_Contact_Case_Summary_ContactInformation and name it CMGT_[NameOfEntityType]_Case_Summary
    ii. Ensure the desired entity type fields are displayed
    iii. Make sure that it takes in the rule input of entity to match CMGT_Case_Summary_EntityTypeContactInfo where it will be displayed
    iv. CMGT_Case_Summary_EntityType
        1. In the match statement add a new "equals, then" parameters with the constant that points to the Entity Type Id and the new interface to display

b. If it's desired to display details of this new entity type in the entity record summary view
    i. Duplicate CMGT_Entity_Person_Contact_Summary_Info and name it CMGT_Entity [NameOfEntityCategory]_[NameOfEntityType]_Summary_Info
    ii. Ensure the desired entity type fields are displayed
    iii. Make sure that it takes in the rule input of entity to match CMGT_Entity_Summary_Information_EntityType_Match where it will be displayed
    iv. CMGT_Entity_Summary_Information_EntityType_Match
        1. In the match statement add a new "equals, then" parameters with the constant that points to the Entity Type Id and the new interface to display

6. Miscellaneous Updates (optional)

The rules below are optional to update:

- CMGT_UTIL_FormatPersonEntityAddress
  - This could be included as an example of a miscellaneous formatting rule that may be desired, but it's entirely optional
- CMGT_Portal_Email_EmailSettingMap_CreateCase
  - Email addresses are used here and any new types which should be sent emails should be added here
- CMGT_AA_RuleCondition_evaluationForEntityState / CMGT_AA_RuleCondition_evaluationForEntityPostalCode / CMGT_AA_RuleCondition_evaluationForEntityCity
  - Similar disclaimer to CMGT_Portal_Email_EmailSettingMap_CreateCase - if you want to update for a specific entity that has an address, you can

Associate Entity on case creation

Out of the box, entities can only be associated with cases after the case is submitted. Customers may want the submitter to select entities while creating the case. To enable this:

1. Edit the expression rule CMGT_Case_ReturnListOfStepsForCreateCase
   a. Add a new map for step 2:
```
a!map(
    formIndex: 2,
    label: "Relate Entities",
    showWhen: true(),
    showBaseButtons: true
  ),
```
   b. Increment the original step 2 to 3
2. Import the package [CMGT Case Management Studio Base - associateEntityOnCaseCreate - 2024-02-23_0031]- this contains:
   a. a new interface to associate entities
   b. a new record grid interface that is compatible with portals
3. Edit the interface CMGT_Case_Sub_Create
   a. Add a rule input 'relatedEntities' of type CMGT_CaseEntityMap, Click the Array checkbox
   b. In the match conditional, add logic to display the following for step 2:
```
equals: 2,
            then: a!sectionLayout(
              label: "Relate Entities",
              labelSize: "MEDIUM_PLUS",
              labelColor: ri!brandingMap.StandardTextColor,
              labelHeadingTag: "H2",
              contents: {
                a!sectionLayout(),
                rule!CMGT_AssociateEntityToCaseOnCreate(
                  case: ri!case,
                  caseEntityMaps: ri!relatedEntities,
             brandingMap: ri!brandingMap
                )
              },
            ),
```

4. Edit the interface CMGT_Case_CreateCase_FormDisplayWrapper
    a. Add a rule input 'relatedEntities' of type CMGT_CaseEntityMap, Click the Array checkbox
    b. Update match statement logic such that is the value =1 or value=2 then use the Submit case form
    c. In the call to CMGT_Case_Sub_Create(), add the parameter
       `relatedEntities: ri!relatedEntities`
    d. Increment the value for the CMGT_CB_CreateCase_WizardDisplay to 3
5. Edit the interface CMGT_Case_RecordAction_Create
    a. Add a rule input 'relatedEntities' of type CMGT_CaseEntityMap, Click the Array checkbox
    b. In the call to the process start form CMGT_Case_CreateCase_FormDisplayWrapper(), add the parameter
       `relatedEntities: pv!relatedEntities`

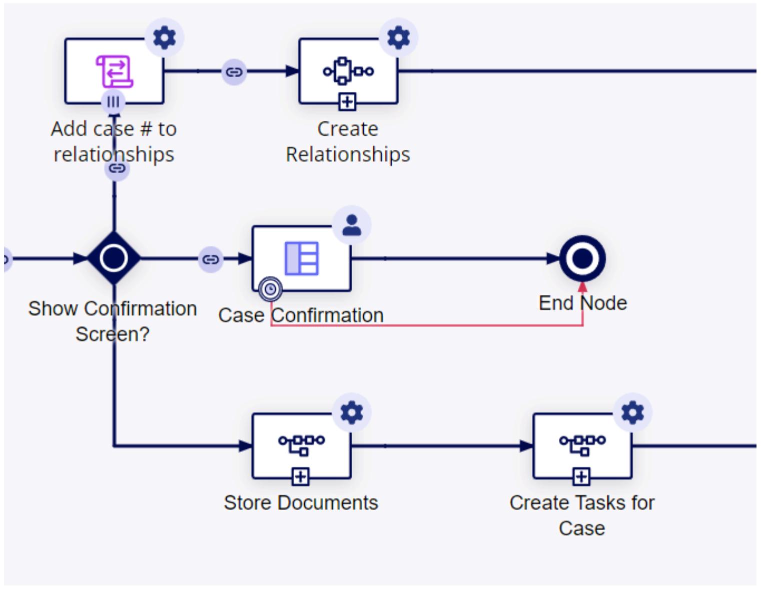
6. Edit the interface CMGT_Portal_Sub_CreateCase
    a. Add a rule input 'relatedEntities' of type CMGT_CaseEntityMap, Click the Array checkbox
    b. In the call to CMGT_Case_Sub_Create(), add the parameter
       `relatedEntities: ri!relatedEntities`

7. Edit the interface CMGT_PortalPage_SubmitCase
    a. Add a rule input 'relatedEntities' of type CMGT_CaseEntityMap, Click the Array checkbox
    b. In the call to CMGT_Portal_Sub_CreateCase(), add the parameter
       `relatedEntities: ri!relatedEntities`

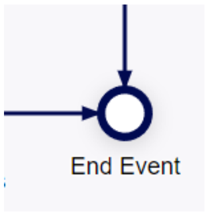
8. Edit the expression rule CMGT_Case_CreateCaseButtons
    a. Add a rule input 'relatedEntities' of type CMGT_CaseEntityMap, Click the Array checkbox
    b. In the call to a!startProcess update the processParameters to include relatedEntities and set value as ri!relatedEntities (ignore invalid process parameter error, will be created in step 11)
       ```
       a!startProcess(
                   processModel: cons!CMGT_PM_CREATE_CASE,
       ```

```
processParameters: {
  case: ri!case,
  relatedEntities: ri!relatedEntities
},
```

9. Edit the interface CMGT_Case_Sub_Create
    a. In the call to CMGT_Case_CreateCaseButtons() when defining local!buttons add the parameter
    `relatedEntities: ri!relatedEntities`
10. Edit the interface CMGT_Case_RecordAction_Create
    a. In the call to CMGT_Case_CreateCaseButtons() when defining local!buttons add the parameter
    `relatedEntities: ri!relatedEntities`
11. Edit the integration CMGT_INT_POST_Portal_CreateCase
    a. Add a new rule input called relatedEntities
    b. In the request body, pass ri!relatedEntities wrapped in a!studio_recordToJson
12. Edit the web api CMGT_API_Portal_CB_CreateCase
    a. Add logic to parse the relatedEntities json
    b. In the process parameters for CMGT_PM_CREATE_CASE, pass related entities
13. Update CMGT_CB_CreateCase_WizardDisplay
    a. Add a rule input for related entities and pass into CMGT_INT_POST_Portal_CreateCase
14. Edit the process model Submit Case to add storing the related entities.
    a. Add a process variable named relatedEntities, of type CMGT_CaseEntityMap (Record Type), multiple, parameter
    b. On the process start form tab, refresh the parameters to be sure the relatedEntities ri! Is properly mapped to the pv!relatedEntities
    c. After the "Show Confirmation Screen" OR node, insert two nodes (Script Task, Subprocess) as depicted below: Be sure to mark all lines as being chained:



d. Along with the existing conditions of the OR gateway, add the below condition:
   =a!isNotNullOrEmpty(pv!relatedEntities)
   i. If the condition is true, go to "Add case # to relationships"
e. Configure the script task as follows:
   i. Other tab:
      1. Run one instance for each item in relatedEntities
      2. Run all instance at the same time
      3. Move on when: all instances are done
   ii. Data tab, create new custom output
      1. Expression: pv!case[CMGT_Case.fields.caseId]
      2. Target: relatedEntities.caseId
f. Configure the subprocess node to invoke the process CMGT_CaseEntityMap_Add. The single parameter is pv!relatedEntities
g. Make sure the subprocess node leads to the End Event node



h. Save and publish process model

Add Task Types

Task types are different variations of either attended (human completed) or unattended (automated) activities that are present in a case's workflow. Instructions for creating new task types can be found in the Case Management Studio documentation at https://docs.appian.com/suite/help/latest/cms-configure-tasks.html.

Add Business Groups and Adjust Security & Permissions

Case Management Studio apps use a combination of groups and object security to assign permission levels to users, which can only be configured in Designer. You can learn about the default groups and their permissions in the Case Management Studio documentation at https://docs.appian.com/suite/help/latest/cms-security.html.

To create new groups follow these steps:
1. Create new Appian groups that best correspond to the organizational breakdown of your use case. Ensure these groups are set to "Public" visibility.
2. Navigate to CMGT Business Groups.
3. Add any newly created groups to this group as direct members - in doing so, these groups will be available for selection in group pickers within CMS.

In general, CMS leaves much of the decisioning around security and permissions up to the implementation team. This is a result of direct feedback from Customer Success and partners working on previous Appian solutions, indicating that it’s much easier to code in security using the Appian platform than it is to rip out existing security settings. Furthermore, every organization has unique requirements around group structures, security, and permissions to the point that this area is best left as a customization.

Using a combination of the newly created business groups, or other groups within the system, we encourage you to leverage the power of the Appian platform to code in security and permissions to best match your case management use case. You can also take advantage of platform features like record-level, record view, and record action security.

Adjust Case Intake Data

CMS supports multiple different case types so that data and workflow can be distinct to best adapt to an organization’s varied use cases.

By default, CMS ships with a case intake form, presented in a modal dialog as a multiple step wizard. Default sections include case types, case details, case documents, and review. More details on how each of these sections ship out of the box are below:

1. “Case Type” contains a selection screen to select the case type from a dropdown.
2. “Details” contains inputs for case title and description.
3. “Documents” contains a document upload field for case creators to upload one or more documents to the case.
4. “Review” is the final step in the case intake wizard which allows the case creator to review all of the inputted data on one page before they submit the case.

We first recommend adding your custom case intake data via the front-end capabilities of the Studio site, and then further extending via Appian Designer as your use case permits.

Common Configurations

Setup

Instructions for setting up Case management Studio can be found in the Appian documentation at https://docs.appian.com/suite/help/24.3/installing-cms.html.

Modify Brand & Images

The Solutions Hub site is used to change the branding and imagery displayed throughout CMS's authenticated user sites.

To access the Solutions Hub navigate to the sites menu, and select "Solutions Hub". Detailed instructions on how to use the Solutions Hub to make brand, image, or text changes can be found at https://docs.appian.com/suite/help/latest/sol-custom-suite-user-guide.html.

There are also some aspects of the public portal that need to be modified directly in that object since these areas are not expression-backed. Navigate to the CMGT_Portal object to adjust the header and branding sections.

Modify Text and Languages

Case Management Studio uses translation sets for all end user text, making it easy to update strings to fit your business terminology or import translations for other locales. For example, if you refer to a "case" as a "request" or a "ticket", you can search in the translation set to find all the places where "case" is mentioned and update them to use your preferred terms.

Here are examples of how your terminology could map to out-of-the-box terms:

Out-of-the-box term	**You may refer to this as**
Case	request, ticket, application, etc.
Task	action item, work item, etc.
Default SLA	work estimate, completion time, expected duration, etc.
Contact	owner, POC, etc.

By default, Case Management Studio is available in English (US). You are responsible for translating CMS to additional locales as needed using your preferred vendor or service. It's recommended to make all vocabulary updates before translating for efficiency.

Create Case Categories & Types

The Studio module consists of an Appian site called "Studio". It's where business users can create case categories & case types. This is also where you can configure custom fields for case types and categories, create tasks and workflows, and customize case intake forms.

Instructions on using the Studio module can be found in the Case Management Studio documentation at https://docs.appian.com/suite/help/24.3/cms-studio-overview.html.

Create Charts & Reports

The Report Builder module consists of an Appian site. It's used to create reports and dashboards of reports in "no-code" that can be shared with the relevant groups.

To access the Report Builder site, navigate to the sites menu and select "Report Builder". Alternatively, we also provide this capability right from the CMS "Workspace" site under "Reports".

Instructions on how to use Report Builder can be found at https://docs.appian.com/suite/help/23.4/cms-amd-eur-module-overview.html.