Search documentation Search

Appian

Referencing System Data Types Versus Custom Data Types

Appian 7.7

In Appian, data objects must have a data type that defines the object. Data types fall into two categories, system and custom.

Toggle Us

System Data Types

System types are the standard data types that ship with Appian. They include primitive data types such as *Text* and *Number* — as well as Appian Obj types such as *Document* and *People*. All system types are predefined. Their structure cannot be modified or deleted. System data types only include single type of data. They can contain either a single value or multiple values.

Custom Data Types

Custom data types are created as part of a process that uses the call web service node (when importing a WSDL) or by a system administrator who uploads a schema definition (XSD). Uploaded schemas must adhere to the w3c specification.

Tu

Unlike system data types, custom data types can be deleted. The structure of a custom data type can also be redefined by deleting the data type ar re-importing the data type definition with the same name and namespace.

Selecting a Data Type

Use a **system** data type:

- When a process is expected to be long-lived.
- If you make frequent in-flight changes to process data.
- If your variable is expected to contain large amounts of text.

Re

Use a **custom** data type:

- When interfacing with third party systems.
- When a process is expected to be of a short duration.
- When sharing data between processes and sub-processes.
- When using data stores and query rules.

Guidelines

The following guidelines may provide you with useful assistance when planning the design of the data structures used by your process applications.

Rules and Constants

Ot

Object	System Data Type	Custom Data Type
Rules	Supported as parameters to Rules	Not supported as parameters to Rules
Constants	Supported	Not supported as constants

Process Execution

Usage	System Data Type	Custom Data Type
Default values for variables (PVs and Node Inputs)	Default value defined using an expression or literal	Default value defined using an expression (for Node Inputs, values can be assigned to fields)
Update process variables in-flight	Supported	Use a script task to update the variable.
Used in expressions...	`pv!mySimplePV`	`pv!myCustomPV`
Using fields in expressions...	N/A	`pv!myCustomPV.fieldname`
Setting values using expressions...	pv!mySimplePV=" text"	pv!myCustomPV = {a:1, b:" text"}
Sizing	10 PVs use the same memory as in 5.6.x	1 Custom PV with 10 fields uses slightly less memory than 10 PVs

Process History

Usage	System Data Type	Custom Data Type
When a value is edited…	A new value is displayed in the history when changed.	The new value of entire CDT is displayed when at least one field changed.
Sizing	The changed value is stored in the process history.	A change to a single field stores the entire process variable again in the process history.

Forms

Usage	System Data Type	Custom Data Type
Form inputs…	Map from a form input to a node input of the same data type.	Map a form input to a custom field
Grids	A multiple-value process variable is required for each column in a grid form component.	A multiple-value custom process variable can be used to capture the input of grid component when each column is mapped to a field.
Displaying values using expressions	Same as process execution	Same as process execution

Reports

Object	System Data Type	Custom Data Type
Report	Supported	Columns can appear blank if the variable doesn't exist for all rows. Accessing a field in a custom data type is a slower operation

Integration

Object	System Data Type	Custom Data Type
Call Web Service Smart Service	Only allows mapping of simple web service inputs and outputs	Allows you to map a wide range of custom web service inputs and outputs
Data Store Entities	Not available	Allows you to persist data using custom data types in an RDBMS.