Incorporate CAD Metadata Into an Experience

This topic provides a full list of supported functions, their arguments, and outputs.

These APIs are based on promise. For more information on promise, see **https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise**.

API Functions

Declaration	Parameters	Description
`get` (idpath, propName, categoryName)	`{string|string[]} idpath`— id path such as `'/0/1'`, or array of id paths `['/0/1', '/0/2']`. `{string|string[]} propName`—(Optional) For example, `'Display Name'` or `['Display Name', 'Part ID Path']` `{string|string[]} categoryName`—(Optional) For example, `'PROE Parameters'`. If `propName` was `string []`, then `categoryName` must also be an array of matching length (or undefined).	Gets a metadata object representing the id path or property value(s) for the given `idpath` and `propName`. This function returns the metadata object representing the given `idpath`, or if `propName` is given then the value of the property on the component. Example: `PTC.Metadata.fromId('model-1').then( (metadata) => { var result = metadata.get('/0/6', 'Display Name') });`
`getProp` (propName, categoryName)	`{string|string[]} propName`—(Optional) For example, `'Display Name'` or `['Display Name', 'Part ID Path']` `{string|string[]} categoryName`—(Optional) For example, `'PROE Parameters'`.	This function returns all string property values from a single component, or undefined if no data/components available. If the given `propName` was an array, it returns string[] of values. Example: `PTC.Metadata.fromId('model-1').then( (metadata) => { var result =`

	If `propName` was `string []`, then `categoryName` must also be an array of matching length (or undefined).	`metadata.get('/0/1').getProp('Display Name');` `});`

`getCategory` (categoryName)	`{string} categoryName`	This function returns object with all property names and values from given category.
		Example:
		`PTC.Metadata.fromId('model-1').then( (metadata) => {`
		`var result = metadata.get('/0/6'). getCategory ('__PV_SystemProperties');`
		`});`

`getSelected` (selectFunc)	`{function} selectFunc`—(Optional) Function that controls the values put into the returned array. The function is given `idpath` and an argument and current metadata as:
	`` `this` function(idpath) { return [idpath, this.get(idpath, 'Display Name')]; }); ``
		This function returns an array of whatever is returned by the given `selectFunc`, or if `selectFunc` is undefined, then it returns string[] of id paths.
		Example:
		`PTC.Metadata.fromId('model-1').then( (metadata) => {`
		`var selectFunc = function(idpath) { return metadata.get(idpath, 'Display Name'); }`
		`var result = metadata.getSelected(selectFunc);`
		`});`

`find` (propName, category)	`{string} propName`—(Required) `{string} category`—(Optional)	Finds components based on property values. Also see `findCustom` below. Returns a finder for components based on given the `propName` and category. Example: `PTC.Metadata.fromId('model-1').then( (metadata)`

		=> {
		var displayName = metadata.find('Display Name').like('BOLT');
		});
		PTC.Metadata.fromId('model-1').then((metadata) => {
		var result = metadata.find('Part Depth').lessThan(3).find('Display Name').like('PRT');
		});
		
		PTC.Metadata.fromId('model-1').then((metadata) => {
		var selectFunc = function(idpath) {
		return metadata.get(idpath, 'Display Name')
		var result = metadata.find('Part Depth').greaterThan(4, selectFunc)
		});
		The comparison can be as follows:
		- startsWith,like,sameAs,unlike : string comparison
		- equal,notequal,greaterThanEq,lessThanEq,lessThan,greaterThan : numeric comparison
		- in,out : numeric range comparison
		- before,after : date/time comparison

findCustom (whereFunc, selectFunc)	{function} whereFunc—(Required) {function} selectFunc—(Optional)	Also see find above. This function returns a finder for components based on custom whereFunc. The following example finds all components with depth<2 or has a name like 'ASM'. Example:

		PTC.Metadata.fromId('model-1').then((metadata) => {
		var whereFunc = function(idpath) {
		const depth = metadata.get(idpath, 'Part Depth')
		const name = metadata.get(idpath, 'Display Name')
		return parseFloat(depth) >= 4 || (name && name.search('ASM') >= 0)
		}
		var result = metadata.findCustom(whereFunc);
		});

Supported Find Operators Functions

- `startsWith` (propValue, selectFunc)
- `not` (propValue, selectFunc)
- `sameAs` (propValue, selectFunc)
- `like` (propValue, selectFunc)
- `unlike` (propValue, selectFunc)
- `equal` (propValue, selectFunc)
- `notEqual` (propValue, selectFunc)
- `lessThanEq` (propValue, selectFunc)
- `greaterThanEq` (propValue, selectFunc)
- `lessThan` (propValue, selectFunc)
- `greatThan` (propValue, selectFunc)
- `in` (min, max, selectFunc)

- `out` (min, max, selectFunc)
- `before` (propValue, selectFunc)
- `after` (propValue, selectFunc)
- `findCustom` (whereFunc, selectFunc)

Usage Examples

```
PTC.Metadata.fromId('model-1').then (metadata) => {

    metadata.get('/0/6', 'Display Name')
    => "BLOWER.ASM"

    metadata.get('/0/6'). getCategory ('__PV_SystemProperties')
    => {Component Name: "BLOWER.ASM", Display Name: "BLOWER.ASM", OL File Name: "", Part Depth: "3", Part ID: "6", …}

    metadata.find('Display Name').like('PRT')
    => {id: "model-1", _friendlyName: "Display Name like PRT", _selectedPaths: Array(26)}

    metadata.find('Display Name').like('PRT').find('Part Depth').in(0,3)
    => {id: "model-1", _friendlyName: "Display Name like PRT AND Part Depth in 0-3", _selectedPaths: Array(10)}

    var meta = metadata.find('Part Depth').greaterThan(4);
    meta.getSelected();
    =>["/0", "/0/1", "/0/1/2", "/0/6"]

    var selectFunc = function(idpath) {
      return metadata.get(idpath, 'Display Name');
    }
    meta.getSelected(selectFunc);
    => ["PISTON.PRT", "PISTON_PIN.PRT", "CONNECTING_ROD.PRT"]

    metadata.find('Part Depth').greaterThan(4).getSelected(selectFunc)
    => ["PISTON.PRT", "PISTON_PIN.PRT", "CONNECTING_ROD.PRT"]

    var selectFunc = function(idpath) {
      return metadata.get(idpath, 'Display Name');
```

```
}
metadata.find('Part Depth').greaterThan(4, selectFunc)
=> ["PISTON.PRT", "PISTON_PIN.PRT", "CONNECTING_ROD.PRT"]

var selectFunc = function(idpath) {return metadata.get(idpath, 'Display Name');}
metadata.find('Part Depth').greaterThan(4, selectFunc)
=> ["PISTON.PRT", "PISTON_PIN.PRT", "CONNECTING_ROD.PRT"]
var selectFunc = function(idpath) {return metadata.get(idpath, 'Display Name');}
metadata.find('Display Name').like('PISTON', selectFunc)
=> ["PISTON.ASM", "PISTON.PRT", "PISTON_PIN.PRT"]


var whereFunc = function(idpath) {
  const depth = metadata.get(idpath, 'Part Depth')
  const name  = metadata.get(idpath, 'Display Name')
  return parseFloat(depth) > 4 || (name && name.search('PISTON') >= 0)
}
var selectFunc = function(idpath) {return metadata.get(idpath, 'Display Name');}
metadata.findCustom(whereFunc,selectFunc)
=>["PISTON.ASM", "PISTON.PRT", "PISTON_PIN.PRT", "CONNECTING_ROD.PRT"]

var selectFunc = function(idpath) {
  return metadata.get(idpath).getCategory('__PV_SystemProperties');
}
metadata.find('Part Depth').greaterThan(4, selectFunc)
=> (3) [{…}, {…}, {…}]
0: {Component Name: "PISTON.PRT", Display Name: "PISTON.PRT", OL File Name: "l-Creo 3D_0_ac-40_asm_5.ol" …}
1: {Component Name: "PISTON_PIN.PRT", Display Name: "PISTON_PIN.PRT", OL File Name: "l-Creo 3D_0_ac-40_asm_6.ol",…}
2: {Component Name: "CONNECTING_ROD.PRT", Display Name: "CONNECTING_ROD.PRT", OL File Name: "l-Creo 3D_0_ac-40_asm_7.ol", …}
```

Usage further Examples /old

```
PTC.Metadata.fromId('model-1').then( (metadata) => {  <HERE YOUR CODE> });

….

PTC.Metadata.fromId('model-1').then( (metadata) => {  // start of the then block

Let disp_name= metadata.get('/0/6', 'Display Name');
                                                    });// end of then block



 PTC.Metadata.fromId(''model-1'').then( (metadata) => {  // start of the then block

    metadata.get('/0/6', 'Display Name')
    //=> "BLOWER.ASM"

    metadata.get('/0/6'). getCategory ('__PV_SystemProperties')
   // => {Component Name: "BLOWER.ASM", Display Name: "BLOWER.ASM", OL File Name: "", Part Depth: "3",
Part ID: "6", …}

    metadata.find('Display Name').like('PRT')
    //=> Metadata {id: "model-1", _friendlyName: "Display Name like PRT", _selectedPaths: Array(26)}

    metadata.find('Display Name').like('PRT').find('Part Depth').in(0,3)
   // => Metadata {id: "model-1", _friendlyName: "Display Name like PRT AND Part Depth in 0-3",
_selectedPaths: Array(10)}

    var meta = metadata.find('Part Depth').greaterThan(4);
    meta.getSelected();
   // =>["/0", "/0/1", "/0/1/2", "/0/6"]

});// end of then block
```