

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Version du document : 2
Version MySurvey : 1.8.9.1

Date : 10/04/2026

Introduction : Propriétés et attributs

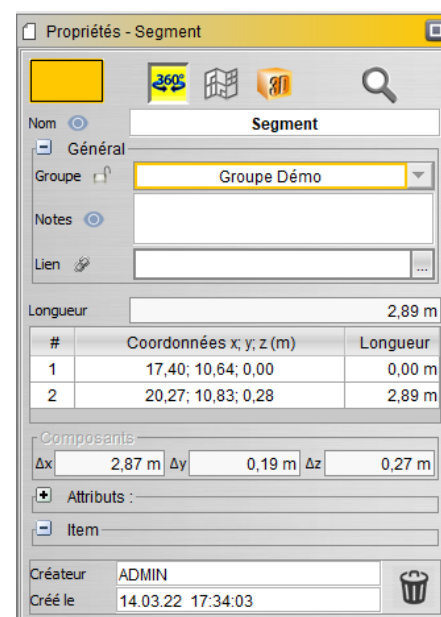
Les éléments et les groupes dans MySurvey sont considérés comme des « fournisseurs de propriétés » car ils stockent une liste définie de *propriétés* en interne.

Une propriété elle-même est définie par une paire *clé/valeur* telle que NAME à « Groupe Démo ».

Les propriétés disponibles et leur accès (lecture/écriture) dépendent du type de fournisseur.

Le Gestionnaire de propriétés (PM) est la fenêtre de panneau dans MySurvey qui affiche toutes les propriétés du fournisseur actuellement sélectionné.

Cela peut inclure des éléments tels que le nom, le groupe parent, les commentaires, les liens vers les pièces jointes, les attributs géométriques, le créateur, la date, etc.

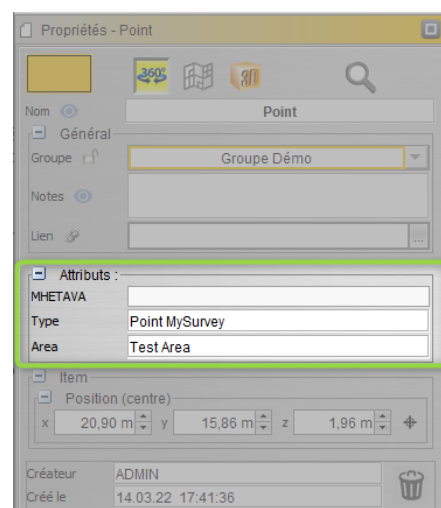


#	Coordonnées x; y; z (m)	Longueur
1	17,40; 10,64; 0,00	0,00 m
2	20,27; 10,83; 0,28	2,89 m

Actuellement, la seule partie du panneau personnalisable est la section intitulée « Attributs ».

Ici, les attributs seront placés lorsque les conditions sont remplies et à l'aide d'un langage de script qui sera décrit plus loin.

Si l'attribut est défini comme éditable, les utilisateurs pourront entrer des valeurs dans les champs et elles seront stockées dans la base de données.



Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Configuration et langage de script

Script de règles

Afin de gérer les propriétés de manière pilotée par les données, le fichier de script **main.rules** peut contenir un ensemble sophistiqué d'instructions (conditions et actions) définissant le comportement personnalisé dans MySurvey.

Les règles peuvent être également définies dans un fichier séparé situé à côté et ayant la même syntaxe. Le fichier **main.rules** devra en ce cas « intégrer » ce fichier à l'aide de la commande

```
import "nom_du_fichier.rules"
```

Propriétés personnalisées (attributs)

Des propriétés personnalisées supplémentaires peuvent désormais également être attachées en tant que métadonnées (également appelées « attributs ») dans MySurvey.

L'utilisateur peut définir une **clé** de chaîne de son choix (ne doit pas entrer en conflit avec les clés intégrées répertoriées ci-dessous) et peut ensuite y attacher une **valeur** de chaîne en tant que métadonnées.

La syntaxe d'une structure d'attributs commence par le mot-clé **new**, se termine par le mot-clé **end** et peut contenir une liste de paramètres facultatifs écrits sous la forme **<clé>=<valeur>**.

Exemple de structure d'attribut pour l'ajout d'une propriété personnalisée unique :

```
new Attribute « TEMPERATURE »
  CAPTION="Temp. °C »
  TOOLTIP="La température interne quotidienne la plus élevée mesurée en
degrés Celsius »
  TYPE_ATTRIBUT=STRING
  EDIT=ENABLED
  BUBBLE=COLLAPSED
end
```

Description de chaque paramètre, dont certains ont plusieurs clés valides en anglais et Français:

Clé	Description du paramètre
CAPTION	Il s'agit du nom de la propriété personnalisée qui sera affiché dans l'étiquette de champ PM, l'en-tête de colonne CSV, etc. Il n'est pas sensible à la casse !
TOOLTIP	Il s'agit d'un texte informatif facultatif (phrase) à afficher lorsque le curseur de la souris survole le champ PM correspondant à cette propriété personnalisée.
TYPE_ATTRIBUT	Type de variable stockant les données, pour l'instant seul STRING est pris en charge.
EDIT	Si la valeur peut être modifiée dans le champ PM. Valeurs possibles: READ_ONLY/ENABLED
BUBBLE	Comment afficher la propriété personnalisée dans la bulle contextuelle : SHOW/HIDE/COLLAPSED

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Importation de propriétés personnalisées (attributs)

Lors du chargement d'un fichier au format RVM, si le fichier ATT facultatif portant le même nom est présent, il sera chargé en même temps que le fichier RVM pour obtenir et fusionner des métadonnées supplémentaires avec les fournisseurs de propriétés, même si le fichier RVM est une sous-arborescence de l'arborescence décrite dans le fichier ATT.

Syntaxe du script

La syntaxe du script est similaire au style de langage C, mais avec des simplifications pour une meilleure lisibilité.

<code>// Commentaire d'une seule ligne</code>	Tout le texte jusqu'à la fin de la ligne est ignoré par l'analyseur.
<code>/* Commentaire multiligne */</code>	Tout le texte entre les jetons de début et de fin est ignoré par l'analyseur.
<code>« chaîne de caractères » « avec signes spéciaux \"escape\" »</code>	Les chaînes de caractères sont entourées de guillemets doubles et peuvent contenir des espaces et des caractères d'échappement à l'aide de la barre oblique inverse \. Ceux-ci sont obligatoires pour les noms personnalisés, les chemins, etc. afin d'éviter les conflits avec les mots-clés.
<code>["el1", "el2", "elN"]</code>	Les tableaux sont un regroupement de plusieurs éléments délimités par des crochets avec des délimiteurs de virgules.
<code>["root"/"parent"/"child"]</code>	Les chemins sont comme des tableaux mais sont délimités par des barres obliques /
<code>methodName("arg1", "arg2", "argN")</code>	Les méthodes ont une liste d'arguments facultative qui ressemble à des tableaux mais délimitée par des parenthèses.

Vous trouverez ci-dessous une liste des mots-clés réservés et leur signification.

Mots-clés réservés	
<code>if, else, end</code>	Mots-clés de structure conditionnelle. Voir les exemples ci-dessous.
<code>and, or, not</code>	Opérateurs conditionnels. Voir les exemples ci-dessous.
<code>equals, in, contains</code>	Opérateurs de comparaison binaire avec résultat booléen.
<code>for, break, continue, end</code>	Mots-clés de structure itérative (<i>pas encore disponible</i>).
<code>return, exit</code>	Réservé (<i>Pas encore disponible</i>)
<code>alias</code>	L'utilisation est : <code>alias dest source</code> . Effectue une simple recherche/remplacement de toutes les instances correspondant au mot <code>dest</code> avec <code>source</code> . Un avertissement sera donné si un alias en double est défini, mais il est légal et remplacera la définition précédente par la nouvelle. Le résultat d'exécution utilise toujours le dernier alias défini.
<code>import</code>	Analyse le fichier spécifié avant de passer à la ligne suivante, comme le fonctionnement de C <code>#include</code> .

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Méthodes intégrées	
<code>prop (type)</code>	Récupère la propriété avec l'identificateur donné. Cet identificateur peut être le TYPE de n'importe quel attribut personnalisé ou l'un des nombreux identificateurs de propriété intégrés. Voir la liste des propriétés.
<code>path (type)</code>	Récupère le chemin d'accès ancêtre (y compris self) de la propriété demandée. Si aucune propriété n'est spécifiée, la propriété <code>NAME</code> par défaut est utilisée.
<code>showInBubble (type, override)</code>	Afficher une nouvelle propriété personnalisée dans la bulle d'info-bulle (ligne de texte)
<code>setAsRemarkable ()</code>	Définir un élément d'assemblage remarquable (qui ne sera pas compacté)
<code>assemble ()</code>	Définir un groupe à assembler automatiquement (lors de l'importation)
<code>addColumnCSV (type, override)</code>	(Pas encore disponible) Afficher une propriété personnalisée dans les fichiers exportés (colonne CSV)
<code>addColumnAssembly (type)</code>	Affichez une colonne supplémentaire dans la liste des assemblages PM.
<code>addImportMeta</code>	Ajout d'une paire clé/valeur de métadonnées lors de l'importation.
<code>showInPM ()</code>	Affiche une propriété personnalisée dans le panneau PM (champ de texte)
<code>propAncestor (prop, PROP_ANCESTOR, VALUE_ANCESTOR_1, ..., VALUE_ANCESTOR_N)</code>	Récupère la valeur de la propriété d'un ancêtre <code><prop></code> . Un ancêtre approprié est déterminé en comparant la valeur <code><PROP_ANCESTOR></code> de l'ancêtre à la liste des valeurs admissibles : <code><VALUE_ANCESTOR_1>, ..., <VALUE_ANCESTOR_N></code>
<code>propAncestorIfUnset</code>	Identique à <code>propAncestor</code> mais récupère d'abord la propriété « self » si disponible.
<code>setFlag (key, value)</code>	Ajoutez (ou remplacez) une paire clé/valeur de chaînes dans la liste des variables globales. Une liste de clés et de valeurs prédéfinies est affichée ultérieurement.

Coloration syntaxique

Pour faciliter la lecture, un fichier `rulesNpp.xml` a été inclus pour une utilisation avec **Notepad++** afin de mettre en évidence les couleurs de syntaxe des mots-clés dans n'importe quel fichier `*.rules`.

Sécurité de type

Très peu de vérifications sont effectuées au moment de la compilation pour s'assurer que les opérandes/arguments sont des types appropriés pour leurs opérateurs/méthodes respectifs. Cela peut entraîner des erreurs d'exécution, mais elles seront accompagnées de rapports d'erreurs détaillés avec les numéros de ligne où les problèmes se sont produits.

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Exemples

Dans cet exemple, deux alias sont utilisés pour faciliter la lecture du reste du code.

Ensuite, en fonction de la propriété personnalisée « TYPE » du fournisseur, il affichera soit la valeur de température dans l'info-bulle, soit dans le PM.

```
alias type prop (« TYPE »)           // Alias pour obtenir le type d'un fournisseur
alias T « TEMPERATURE »             // Alias du nom de la propriété

if type in ["PIPE","HVAC","EQUI"] // Si le type du fournisseur est PIPE, HVAC ou EQUI
  showInBubble(T)                  // alors affiche la température dans l'info-bulle
else
  showInPM(T)                      // Sinon affiche la température dans un champ PM
end
```

L'alias peut être utilisé pour simplifier une représentation de tableau. Les conditions peuvent également être jointes par **and**, **or**, **else if**, etc.

```
alias SUBCOMPS ["BRAN","TUBI","DAMP","SILE","ACCES PANEL"] //alias sur tableau

if type in SUBCOMPS           // Si le type du fournisseur est contenu dans le tableau aliasé
ou type in ["PIPE","HVAC","EQUI"] // Ou Si le type du fournisseur est PIPE, HVAC ou EQUI
  showInBubble(T)              // Alors afficher la température dans l'info-bulle
else if type equals « ATTA »    // Sinon si le type du fournisseur est ATTA
  assemble()                   // compacter pendant l'import
else
  showInPM(T)                  // Sinon, ajoutez un champ de température dans PM
fin
```

Remplacement d'une propriété à l'aide de la valeur d'un ancêtre :

```
propAncestorIfUnset ("NAME", "TYPE", "BRAN", "PIPE", "HVAC")
```

Explication : renvoie la valeur **NAME** du fournisseur actuel s'il a un **NAME** (autre que vide ou « unset »), sinon il recherchera vers le haut en commençant par la clé de propriété de son parent « **TYPE** », qui doit être une valeur de « **BRAN** », « **PIPE** » ou « **HVAC** ».

Si aucun de ces ancêtres n'est trouvé, renvoie un résultat vide.

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Utilisation de chemins de types d'ancêtres dans des conditions complexes.

```
alias pathType path("TYPE") //alias pour obtenir un chemin composé de types d'ancêtres

if type in ["PIPE","HVAC","EQUI"] //Si le type du fournisseur est PIPE, HVAC ou EQUI
    showInBubble(T) //Afficher la température dans l'info-bulle

//Si Le parent et grand-parent direct sont respectivement de type PIPE et BRAN
else if pathType equals ["*"/"PIPE"/"BRAN"/"»?»"] /
    showInBubble(T,propAncestor(T,"TYPE","BRAN","PIPE"))

else if pathType contains ["EQUI"/"?" /"*"] //EQUI doit être un ancêtre (parent ou supérieur)
    showInBubble(T)

else if pathType contains ["BRAN"] //Le type du fournisseur doit être BRAN ou avoir un ancêtre BRAN
and type in SUBCOMPS //Le type du fournisseur doit être répertorié dans le tableau
    showInBubble(T)

else if pathType equals ["*"/"STRU"/"?" ] //Le parent direct doit être de type STRU
    showInBubble(T)
//Le type de fournisseur doit être un EQUI et un avoir comme parent STRU
else if pathType equals ["*"/"STRU"/"EQUI"]
// Le fournisseur doit hériter de la température du premier EQUI ou STRU disponible
    showInBubble(T,propAncestorIfUnset(T,"TYPE","EQUI","STRU"))
end
```

Propriétés exposées de MySurvey :

Les utilisateurs auront accès à un ensemble limité de propriétés intégrées à l'aide de la liste des clés du tableau ci-dessous.

La liste (à élargir) des propriétés dans MySurvey:

Clés de propriété intégrées		
Clé	Accès	Description
Base		
NOM	RW	Nom du fournisseur.
TYPE_MS	R	Le type MySurvey (intégré) du fournisseur. Voir la liste TYPE_MS ci-dessous
Cylindre		
DIAMÈTRE	RW	Diamètre cylindrique
RAYON	RW	Rayon cylindrique
HAUTEUR	RW	Hauteur cylindrique
CENTER_BOTTOM	RW	Le centre de la face circulaire inférieure du cylindre
CENTER_TOP	RW	Le centre de la face circulaire supérieure du cylindre
AREAUV	R	Les rectangulaires du cylindre sont (diamètre x hauteur)
VOLUME	R	Le volume 3D du cylindre
...		

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Global flags/variables

Calling `setFlag(key, value)` will add or overwrite the entry with the specified key (case-sensitive) paired with the corresponding string value. It can be conceptualized as static global variables used within the context of the rules where both the key and value are stored as strings that can be evaluated at run-time.

Pre-defined global variable flag keys and values

Below is a list of the available predefined keys and accepted values and their purpose. Many are "Booleans" which are expecting either "TRUE" or "FALSE" (typically anything other than "TRUE" is also considered false).

Key	Type	Purpose
"SHOW_EMPTY_BUBBLES"	Boolean	When "TRUE", custom properties which have empty string values are NOT shown in the bubble tooltip popup (even if requested by <code>showInBubble</code>).
"PULL_UP_ONLY_CHILD"	Boolean	When "TRUE", after importing a file of 3D models, a post-processing step is performed to recognize and pull-up any child item that is alone within its parent group.
"METADATA_KEY_AS_PROP"	Boolean	When "TRUE", all imported metadata keys will be interpreted as potential custom properties (attributes) and added automatically. They can ALL be shown via wildcard: <code>showInBubble("**")</code> or <code>showInPM("**")</code> which does NOT show the built-in properties.

Personnalisation des attributs et de l'arborescence

Fichier de configuration des règles & langage de Script

Types d'objets « Natifs » de MySurvey:

Il est possible d'accéder aux objets natifs de MySurvey en utilisant les types d'objets définis dans l'alias TYPE_MS défini dans le fichier default.rules : "Annotation 3D", "Box", "Circle", "Cone", "Cylinder", "Dish", "Extruded polygon", "Mesh", "Photo", "Piping", "Point", "Point of view", "Polygon", "Polyline", "Pyramid", "Snapshot", "Sphere", "Rectangle", "Segment", "TextArea", and "Trihedron".

Exemple:

```
if prop("TYPE_MS") equals "Sphere" // Si le type du fournisseur MySurvey 3D est une Sphere
  showInBubble("NAME") // Alors afficher le nom de l'objet dans l'info-bulle
else
  showInPM("NAME") // Sinon afficher le nom de l'objet dans la fenêtre de propriétés
end
```