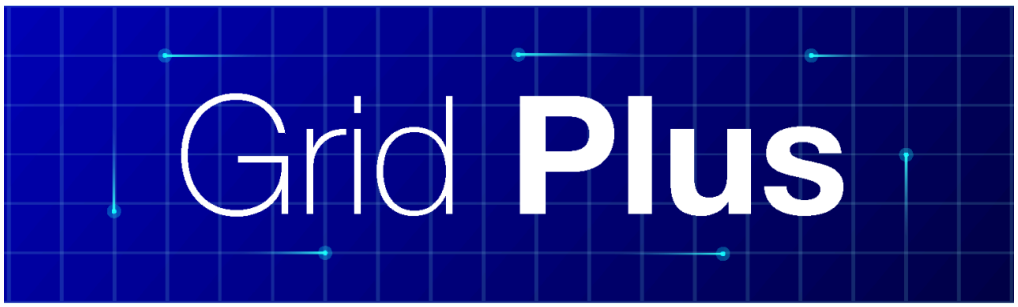


GridPlus Component and Helper Plug-Ins

Advanced Plug-in Migration Guide

Version 1.0.0 | **Advanced Plug-in**

appian

Table of Contents

Overview 2
Version 3.0 to Advanced Plug-in Version 1.0 2
New Features 2
Changes 2
Migration Guidelines 3
Reading & Writing Changes 3
Column Configurations 4
Exporting 6
Styling 6
Additional Parameters to gridPlusLayoutField 6

Overview

This document provides guidance on how to smoothly upgrade your versions of the GridPlus Component and Helper Plug-Ins from Version 3.0 to Version 1.0 of the Advanced Plug-in Program offering, available on Advanced and Premium tiers of Appian.

Version 3.0 to Advanced Plug-in Version 1.0

New Features

- **Reading & writing record data directly from the component**. Designers no longer have to set up complex SAIL code to manually populate the component with data and create buttons for submitting changes and deletions. New records, edited records, and deleted records now call corresponding process models to initiate the write logic, with minimal designer configuration.
- **Security.** The component has new permissioning allowing granular access control over created, edited, and deleted data as well as record and related actions.
- **Pagination.** The grid can now support an entire record type's data with pagination of up to 5,000 records per page**.**
- **Filtering & Sorting.** Users can now perform sorting and filtering of the entire record type, not just the current page of data. Designers can configure the grid with or without a default filter or a default sort.
- **Record & Related Actions.** Users can now create and update records through a traditional Appian Record/Related Action form interface. Additionally, this enables GridPlus to support editing related record data.
- **Theming.** Designers can now customize the theme of the grid, record action interfaces, accent color, button shape, and background color to suit your organization's branding.

- **Selectable cells.** The new onSelect parameter enables users to save the content from selected cells into Appian.
- **New submit changes flow.** Users can now see their changes highlighted in the grid and can review their updates, deletes, and creations from the pending changes modal.
- **Row height.** The new rowHeight parameter allows dynamic and static row heights.
- **Record links.** Record links can now be configured to display both site pages and Tempo interfaces.
- **Dates**. Appian Date type is now supported with the griddatecolconfig() function.
- **Row Headers**. Row headers now correspond to the primary key of the record, for easy data tracking. This can be toggled on or off.

Breaking Changes

- **Grid name.** The component rule name has been changed to **gridPlusLayoutField** and has been reverted to version 1.0.0.
- **ChangeDataValue.** The parameter that tracked all changes in the grid has been separated into three parameters (along with corresponding saveIntos): createDataValue, editDataValue, and deleteDataValue.
- **All record field values.** All designer-configured record fields should be passed into the new **gridRecordFields()** function, instead of using toxml() or tostring(). The recordType itself can be passed to the parameter without any modification.
- **Modified column configuration functions** with revised parameters to directly pass in record types and record fields, and for vertical and horizontal alignment within a column.
- **Removal of the primaryKeyField parameter.** This parameter is no longer necessary.
- **Data Visibility.** Previously, the grid's column visibility was configured by the query. Now, column visibility is explicitly defined by the columnConfigs parameter.

NOTE: More information on these new features can be found in the official GridPlus documentation.

Prerequisites

1. In the Admin Console, remove the previously installed versions for the GridPlus Component and GridPlus Helper plug-ins, and deploy the Advanced Plug-in versions.
2. In the interface that displays the grid component, change the component field rule name, which was previously **gridPlusLayout**, to **gridPlusLayoutField**.

Function Name Changes

All functions in GridPlus now are prefixed with the "grid" keyword. Here are the list of functions that have changed:

- checkboxcolconfig → gridcheckboxcolconfig
- datecolconfig → griddatecolconfig
- dropdowncolconfig → griddropdowncolconfig
- exportconfig → gridexportconfig
- exportrange → gridexportrange
- recordlinkconfig → gridrecordlinkconfig
- numericcolconfig → gridnumericcolconfig
- textcolconfig → gridtextcolconfig
- validator → gridvalidator

Example Setup

```
a!localVariables(
  local!Director_RecordViewLink: gridrecordlinkconfig(
    recordPrimaryKeyField:gridRecordField( GA_Director.directorId ),
    urlStub: "summary"
  ),
  local!Director_RecordUpdateLink: gridrelatedactionconfig(
    recordPrimaryKeyField: gridRecordField( GA_Director.directorId ),
    relatedActionModel: cons!GA_CREATE_DIRECTOR,
  ),
  {
    gridtextcolconfig(
      recordField: gridRecordField( GA_Show.title ),
      linkConfig: null,
      title: "Title",
      validator: gridvalidator(
        name: "titleValidator",
        operator: "isNotNullOrEmpty"
      ),
      readOnly: null,
      colWidth: 250,
      horizontalAlign: "RIGHT",
      verticalAlign: "TOP"
    ),
    gridcheckboxcolconfig(
      recordField: gridRecordField( GA_Show.availableGlobally ),
      label: null,
      labelPosition: null,
      checkedTemplate: null,
      uncheckedTemplate: null,
      title: "Global",
      validator: null,
      readOnly: null,
      colWidth: 130
    ),
    gridnumericcolconfig(
      recordField: gridRecordField( GA_Show.rating ),
      linkConfig: null,
      format: "0.0%",
      title: "Viewer Ratings",
      validator: gridvalidator(
        name: "ratingValidator",
        operator: "greaterThan",
        value: 0.82
      ),
      readOnly: false,
      colWidth: 160,
      culture: null,
      horizontalAlign: "CENTER",
      verticalAlign: "MIDDLE"
    ),
    griddatecolconfig(
      recordField: gridRecordField( GA_Show.releaseDate ),
      dateFormat: "MM-DD-YYYY",
      correctFormat: "test",
      title: "Release Date",
      validator: null,
      readOnly: null,
      colWidth: "AUTO"
    ),
    gridnumericcolconfig(
      recordField: gridRecordField( GA_Show.productionCost ),
      linkConfig: null,
      format: "$0,0.00",
      title: "Production Cost",
      validator: gridvalidator(
        name: "costValidator",
        operator: "greaterThan",
        value: 20000
      ),
      readOnly: false,
      colWidth: null,
      culture: null
    ),
  }
)
```

Reading & Writing Changes

Reading & Pagination

To successfully read the entire record type's data into the component, the **rowsValue** parameter has been replaced with **queryRuleName**. The **queryRuleName** parameter accepts an expression rule that runs a!queryRecordType. Configure **queryRuleName** either with the a rule reference (ex. rule!RULE_NAME) or a string of the rule name (ex. "RULE_NAME). This string name allows the component to run a modified version of the rule as the user clicks

through pagination, sorting, and filtering. While paginating, the component adjusts the start index to get the next batch of data and while sorting and filtering, the component modifies the rule's corresponding sort and filters parameters.

For more details on configuring **queryRuleName()**, please refer to the official GridPlus documentation. Shown below is an example for how this parameter can be configured with this function:

```
Unset
gridPlusLayoutField(
       queryRuleName: rule!GA_queryRule, /* Configure as rule!RULE_NAME (without
calling the function with ()) or as a string with "RULE_NAME" */
)
```

In addition, the expression rule definition **should be defined in this format**:

```
Unset
a!queryRecordType(
  recordType: <recordType>,
  fields: {
  },
  pagingInfo: a!pagingInfo(
    ri!pagenumber,
    ri!batchsize,
    ri!sortingInfo
  ),
  fetchTotalCount: true()
/* Optionally configure a default sort or default filters */
)
```

Writing
The **submitChanges** parameter is triggered when the user clicks the "Submit Changes" button and calls the **gridsubmitchanges()** to trigger the appropriate process models.
Along with these, two local variables must be created to ensure that the **gridsubmitchanges**() function is triggered when changes are submitted from the component. This is demonstrated in the SAIL snippet below. In addition, **developers must set a process model for creating,**

updating, and deleting records records. The process model IDs will be utilized in the **gridsubmitchanges()** function.

For more information on the **submitChanges** parameter and the **gridsubmitchanges()** function, please refer to the official GridPlus documentation.

```
Unset
a!localVariables(
	local!submitChanges: "",
	local!createData,
	local!editData,
	local!deleteData
	gridPlusLayoutField(
		submitChangesValue: local!submitChanges,
		submitChangesSaveInto: a!save(
			target: local!submitChanges,
			value: gridsubmitchanges(
				queryRuleName: <query rule name>,
				colConfigRuleName: <column config rule name>,
				createData: local!createData,
				editData: local!editData,
				deleteData: local!deleteData,
				createModel: <process constant>,
				editModel: <process constant>,
				deleteModel: <process constant>
			)
		)

		)
	)
```

In addition to the above, be sure to utilize the **gridrecordfield()** function when configuring the **hiddenFields** component parameters. An example is shown below. The gridrecordfield() function accepts a record field as a parameter. For more information, please refer to the official GridPlus documentation.

```
Unset
hiddenFields: {gridrecordfield(<recordField>), gridrecordfield(<recordField>)}
```

Column Configurations

The column configuration functions have been adjusted to support new parameters and enable developers to directly pass in record types and record fields. **As in the previous version, column configurations function parameters must be in the order provided. Any parameters that are not necessary to your configuration can be marked as null.**

To start, the **recordLinkConfig()** function has been renamed to **gridrecordLinkConfig()**. The function has been modified to include the following changes:

- **recordType** (replacement for recordTypeXml): Enables the developer to pass in the Record Type directly instead of wrapping it inside of a **toxml()** call. Use the new **gridrecordfield**() function which returns a record field and its associated relationship(s).
- **identifierField** (replacement for identifierFieldName)**:** This field is now set to a call to
- **urlStub:** The URL stub for the record view that the link will open. Go to the record type and click Views to find the URL stub for each record view. Default is "summary" if set to null.
- **targetLocation:** The location where the record view should open, which can be either a site page or Tempo. To reference a site page, use the following structure: site![site name].pages.[page web address identifier]. values: "TEMPO" or a reference to a site page. Default: "TEMPO"
- **openLinkIn:** Determines how the record link should open, either a new window or new tab.

An example is shown below:

```
Unset
gridrecordlinkconfig(
   recordType: recordType!<recordType>,
   identifierField: gridrecordfield(<recordField>),
   urlStub: <urlStub name>,
   targetLocation: <target location>
 )
```

For the remainder of the column configuration functions (gridcheckboxcolconfig, griddatecolconfig, griddropdowncolconfig, gridnumericcolconfig, gridtextcolconfig) apply the following changes:

- Change the **field** to **recordField**, and set it to a call to the **gridrecordfield()** function.
- Two additional parameters have been added for styling within a column. These include:
    - **horizontalAlign**, which determines the horizontal alignment

    - **verticalAlign**, which determines the vertical alignment
- Remove the **relationshipName** field. Any information related to a related field is handled by the **gridrecordfield()** function.

An example is shown below:

```
Unset
gridtextcolconfig(
      recordField:
gridrecordfield(<recordField.relationsips.relationship.fields.field>),
      linkConfig: <link config>,
      title: <title>,
      validator: <gridvalidator>,
      readOnly: <readOnly>,
      colWidth: <colWidth>,
      horizontalAlign: "CENTER",
      verticalAlign: "MIDDLE"
    )
```

For more details on the **gridrecordfield()** function, please refer to the official GridPlus documentation.

Exporting

The **exportrange()** function has been renamed to **gridexportrange()**. The **exportconfig()** function has been renamed to **gridexportconfig()**. For the **gridexportrange()** function, which is used inside of **gridexportconfig,** be sure to replace the **startFieldName** and **endFieldName** with calls to **gridrecordfield()**, as shown in the example below:

```
Unset
gridexportconfig(
        exportrange: gridexportrange(1, gridrecordfield(<record field>), 2, gridrecordfield(<record
field>))
)
```

The **exportrange()** function has also been updated to receive a 1 based index, ranging from 1 to 5000 with an exclusive stop. This **exportrange()** is restricted to encompass only one page of data at a time.

Styling

The style component parameter is used to apply styling to the grid component. Previously, the style parameter was set to accept a dictionary. Now, set the style parameter to a **gridStyling()** function call, as shown below. For more details about the **gridStyling()** function, please refer to the official GridPlus documentation.

```
Unset
style: gridStyling(
        theme: "DEFAULT",
        accentColor: "#6cd66c",
        validationColor: "#FF3C00",
        buttonShape: "ROUND",
        headerAlign: "LEFT",
        backgroundColor: "#FFFFFF",
        embeddedInterfaceThemeIdentifier: "example_id" /* This is used to set
the theme of embedded record and related actions. This can be configured in the
Admin Console under Embedded Interfaces*/

      )
```

Additional Parameters to gridPlusLayoutField

In addition to **submitChangesValue** and **submitChangesSaveInto**, additional parameters have been added to gridPlusLayout to support additional features:

- **recordActions:** To import record actions into the Grid component, set this parameter to a **gridrecordactionconfig()** function call, passing in a record action process model constant and the record action name.
- **rowHeight:** An integer value that applies to all rows of the Grid component. To set the row heights dynamically, set this property to "AUTO".
- **onSelection:** Enables users to save selected cells in the Grid component to local variables.
- **enableCreate** - Allows users to create data from the "New Entries" modal and in the creations section of the "pending changes" modal.
- **enableDelete** - Allows users to delete data in the grid and in the delete section of the "pending changes" modal.
- **enableEdit** - Allows users to edit data in the grid and in the edit section of the "pending changes" modal.

Shown below is a code snippet of these new parameters. Please refer to the official GridPlus documentation for more details.

Unset

```
a!localVariables(
	local!onSelectionItems,
	enableEdit: true(),
	enableCreate: false(),
	enableDelete: false(),
	gridPlusLayoutField(
		recordActions:{
			gridrecordactionconfig(
				recordActionModel: <process model uuid>,
				recordActionName:<name of record action>
		},
		rowHeight:"AUTO" ,
		onSelection: local!onSelectionItems
	)
)
```