ANSYS
19
RELEASE
©1971-2017 ANSYS, Inc.
All Rights Reserved.
Unauthorized use, distribution or duplication is prohibited.

ANSYS ACT 19.0 Known Issues and Limitations

Copyright and Trademark Information

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2008 companies.

U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

See the legal information in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, contact ANSYS, Inc.

Published in the U.S.A.

Table of Contents

ANSYS ACT 19.0 Known Issues and Limitations

This document lists known issues and limitations in the ACT 19.0 release.

Limitations on ACT postprocessing of Mechanical results

If you use ACT for postprocessing Mechanical results, you should be aware of the following limitations:

Scoping for custom results

Custom results do not support using a geometric path as scoping. You can only use a selection of nodes and elements as scoping.

Results not available with ACT postprocessing API

The ACT postprocessing API can access only results stored in the result (RST) file. The non-RST file results are listed in the table below and defined with their result code or component labels.

Result Code	Description
SERR	Elemental structural error energy
TERR	Elemental thermal error energy
BERR	Elemental magnetic error energy

Component Label	Description
SUM	Vector sum
MSUM	Mass sum
SFSUM	Surface force sum
1, 2, 3	Principal
INT	Intensity
EQV	Equivalent
MAXSHEAR	Maximum shear
VECTORS	Vectors
WRAPPING_FACTOR	Wrapping factor

ACT Debugger is not currently supported on Linux platform or from AIM

Due to technology differences, the **ACT Debugger** is not currently supported on the Linux platform or from AIM. In 19.0, it is supported only on the Windows platform from the **Project** page in Workbench and from DesignModeler and Mechanical.

ACT is unable to create a chart from Mechanical

When using ACT to create a figure from the chart API, the following error prevents the graphics display in the Mechanical window:

```
Object reference not set to an instance of an object.
```

As a workaround, add the following code to your script to create an empty window in which the chart can display:

```
import clr
clr.AddReference("Ans.UI.Toolkit")
clr.AddReference("Ans.UI.Toolkit.Base")
import Ansys.UI.Toolkit
if Ansys.UI.Toolkit.Window.MainWindow == None:
   Ansys.UI.Toolkit.Window.MainWindow = Ansys.UI.Toolkit.Window()
```

Graphic API Issues in DesignModeler and Mechanical when no extensions are loaded

There are some limitations on the Graphic API in Mechanical and DesignModeler when no extensions are loaded. For instance, the Factory2D does not work. It is therefore advised to have one or more extensions loaded prior to the usage of the Graphic API.

Starting the ACT Start Page on Linux can cause an unexpected shutdown

When you start the **ACT Start Page** on Linux, an unexpected shutdown can occur. On some Linux operating system variants such as Red Hat, removing the package `totem-mozplugin` resolves the issue:

```
yum remove totem-mozplugin
```

On Linux, using the ACT API to access `ElementNode` results always returns 0

On Linux, when the ACT API is used to access `ElementNode` results in ANSYS Mechanical, **0** is always returned.

Limitation on localization support

Localization of ACT is limited to the languages currently supported in ANSYS Workbench. This limitation does not apply to the ability to manage various languages within the extension. For example, the property names created by an extension do not have to be in the same language as the current activated language in ANSYS Workbench.

There is no mechanism to integrate localization for the property names defined by an extension. To manage different languages for your property names, you must develop localization yourself. Both regional settings based on the "." or the "," decimal symbol are available. However, the implementation of the extension should use the "." symbol for any value defined at the XML or IronPython level.

Automation API UI thread and performance

When you are using the ACT automation API, automation methods and properties must be called in the UI thread. Before 18.1, error messages or unexpected shutdowns were the potential results of the call of the automation API from a callback that was not executed in the UI thread.

As of 18.1, this issue is solved by automatically redirecting every call to the automation API in the UI thread. While you will no longer experience error messages or unexpected shutdowns, automatic redirection can potentially decrease performance. Poor performance is most likely to occur when you repeatedly call the automation API many times, such as when looping through nodes. If you notice a significant decrease in performance, you can speed up the process by using one of the following commands to bypass the multiple switches to the UI thread:

- `ExtAPI.Application.InvokeUIThread(func, arg) // This executes the function func with the argument arg in the UI thread.`
- `ExtAPI.Application.InvokeUIThread(func) // This executes the function func in the UI thread.`

A usage example follows:

```
def get_all_object_names(name_list):
    """ Returns the given list populated with all the names of the
objects in the tree. """
    for obj in ExtAPI.DataModel.Tree.AllObjects:
        name_list.Add(obj.Name)
    return name_list
```

Important: Due to technical reasons, the namespace of the automation API object has changed. Therefore, you should not use the method `GetType()` as a way to verify the nature of the object that you have. Instead, you must use the property `DataModelObjectCategory`, which is available on all objects. It returns an `enum` value indicating the type of the object. For example, the following command returns true:

```
ExtAPI.DataModel.Project.DataModelObjectCategory ==
DataModelObjectCategory.Project
```