How to Contact dSPACE

Mail: dSPACE GmbH
Rathenastraße 26
33102 Paderborn
Germany
Tel.: +49 5251 1638-0
Fax: +49 5251 16198-0
E-mail: info@dspace.de
Web: http://www.dspace.com

How to Contact dSPACE Support

To contact dSPACE if you have problems and questions, fill out the support request form provided on the website at http://www.dspace.com/go/supportrequest.

The request form helps the support team handle your difficulties quickly and efficiently.

In urgent cases contact dSPACE via phone:
• General Technical Support: +49 5251 1638-941

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/support for software updates and patches.

Important Notice

This document contains proprietary information that is protected by copyright. All rights are reserved. The document may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the document must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© Copyright 2013 by:
dSPACE GmbH
Rathenastraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

AutomationDesk, CalDesk, ConfigurationDesk, ControlDesk, SCALEXIO, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.
Contents

Migrating Python Scripts from Python 2.5 to Python 2.7 5
Overview of the Main Changes ...............................................................6
Main Changes in Python 2.7 .................................................................6
Main Changes of Python Handling in dSPACE software ...........7
General Information on Using Python Installations .................11
Required Modifications in Your Scripts ............................................12
Migrating Changes in the Python Core Package .................12
Migrating Changes in the PyWin32 Package .........................15
Using the dSPACE Library ..............................................................16
Enhancements to the Standard Python 2.7 Distribution .........17
Migrating Python Scripts from Python 2.5 to Python 2.7

Objective

To migrate Python scripts from Python 2.5 to Python 2.7, you have to note the following points.

If you want to migrate from an earlier version of Python to Python 2.7, you also have to note the migration steps described, for example, in the AutomationDesk Guide or the ControlDesk Next Generation MCD-3 Automation Guide from earlier dSPACE releases.

Where to go from here

Information in this section

| Overview of the Main Changes           | 6 |
| Required Modifications in Your Scripts | 12 |
Overview of the Main Changes

Objective

Gives you information on the main changes in Python 2.7 and the new dSPACE handling of Python.

Where to go from here

Information in this section

| Main Changes in Python 2.7 | 6 |
| Main Changes of Python Handling in dSPACE software | 7 |
| General Information on Using Python Installations | 11 |

Main Changes in Python 2.7

Objective

Provides information on Python 2.7 and the reasons why dSPACE decided to use it.

What’s New documentation from the Python Software Foundation

The following What’s New documents for Python versions from Python 2.5 to Python 2.7 are available from the Python Software Foundation:

- What’s New from Python 2.5 to 2.6
  http://docs.python.org/2.6/whatsnew/2.6.html
- What’s New from Python 2.6 to 2.7
  http://docs.python.org/2.7/whatsnew/2.7.html

You can refer to the Python documentation provided by the Python Software Foundation for further information on migrating Python scripts from Python 2.5 to Python 2.7.

Main reasons for using Python 2.7 instead of 2.5

The main reasons for migrating to Python 2.7 are:

- Support of new operating systems
- Support for newer versions of extension modules.
- Picking up bug fixes
Why not migrate to Python 3.x?

dSPACE decided to use Python 2.7, because of the announced long-term support for running production systems based on Python 2.x. Python 3.x comes with a lot of incompatible changes that would greatly increase the amount of work needed for migration. Besides not all extension modules required by dSPACE software are available for Python 3.x.

Main Changes of Python Handling in dSPACE software

Objective
Migrating to Python 2.7 also changed the Python handling in dSPACE software.

Changed installation of Python 2.7
Up to dSPACE Release 2013-A, the Python 2.5 installation distributed on the dSPACE DVD was a dSPACE setup with specific enhancements required for working with dSPACE software. It was installed to C:\Program Files (x86)\Common Files\dSPACE\Python25.

With Python 2.7, the Python installation on the dSPACE DVD is using the setups for the Python core and extension modules that are provided by the Python community. dSPACE-specific components are now installed as add-ons. This allows you more flexibility when using Python, for example, you can add and update Python packages without affecting the dSPACE installation. However, the dSPACE software will issue a warning if you use a package or package version that is not tested by dSPACE. If you install the Python 2.7 installation from the dSPACE DVD, it is installed on C:\Program Files (x86)\Python27 by default. You can also use Python 2.7 installed in another folder. The dSPACE installation detects an already installed Python 2.7 and checks whether the Python core and required extensions must be installed or updated.
You have to check whether the methods in your scripts to get the Python installation path must be adapted. For example, you can read the installation path from the Windows registry.

- Using 32-bit Windows operating system:
  \texttt{HKLM\SOFTWARE\Python\PythonCore\2.7\InstallPath}
- Using 64-bit Windows operating system:
  \texttt{HKLM\SOFTWARE\Python\Wow6432Node\PythonCore\2.7\InstallPath}

The Python 2.5 installation had its own online help in dSPACE HelpDesk. With Python 2.7, there is no Python user documentation in dSPACE HelpDesk. You have to open the standard user documentation, for example, via \texttt{Start – All Programs – Python 2.7}.

You can modify the Python installation more freely than before, but you have to guarantee that the dSPACE software works. It is therefore recommended not to remove Python packages from the Python distribution installed by dSPACE.

The Python 2.7 distribution on the dSPACE DVD provides the following Python components.

<table>
<thead>
<tr>
<th>Python Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python Core</td>
<td>2.7.5</td>
</tr>
<tr>
<td>PyWin32</td>
<td>2.10.18</td>
</tr>
<tr>
<td>Numpy</td>
<td>1.7.1</td>
</tr>
<tr>
<td>Matplotlib</td>
<td>1.2.1</td>
</tr>
<tr>
<td>WxPython</td>
<td>2.9.4.0</td>
</tr>
<tr>
<td>Py2exe</td>
<td>0.6.9</td>
</tr>
<tr>
<td>Comtypes</td>
<td>0.6.2</td>
</tr>
<tr>
<td>PIL</td>
<td>1.1.7</td>
</tr>
</tbody>
</table>
Usage of Python interpreter changed

Some dSPACE products previously used an embedded Python interpreter. With dSPACE Release 2013-B, these products now use the Python interpreter provided by the Python 2.7 installation. This interpreter offers the same functionality as the embedded interpreter, but some API calls behave differently. See the migration instructions for details.

The following dSPACE products use the Python interpreter:
- ControlDesk Next Generation
- ConfigurationDesk
- ModelDesk
- SYNECT

With this interpreter you can access the Python implementation of the following dSPACE products, if installed:
- dSPACE Python Extensions (for example, the Python modules of the Platform API Package)
- Real-Time Testing
- dSPACE CAN API Deliverables

Due to the fact that the Python interpreter is separated from the dSPACE products, it closes when you shut down Windows even if the dSPACE product is still running. You have to restart the Python interpreter if the dSPACE product requires access to it.

AutomationDesk uses an embedded interpreter. With this interpreter, you can access only dSPACE Python modules that are included in the AutomationDesk installation, such as rtplib2 and matlablib2.

Discontinued dSPACE Python modules

When migrating your Python scripts to Python 2.7 and dSPACE Release 2013-B, note that some dSPACE Python modules have been discontinued.

The following dSPACE Python modules have been discontinued in dSPACE Release 2013-B:
- rtplib
- matlablib
- rs232lib
- failuresimlib
In dSPACE Release 2013-B, ControlDesk 3.x has also been discontinued. Scripts for automating ControlDesk 3.x can be migrated to be used with ControlDesk Next Generation. For the required migration steps, refer to *ControlDesk Next Generation Migration of ControlDesk 3.x Automation*. 

---

**Discontinuation of ControlDesk 3.x**

In dSPACE Release 2013-B, ControlDesk 3.x has also been discontinued. Scripts for automating ControlDesk 3.x can be migrated to be used with ControlDesk Next Generation. For the required migration steps, refer to *ControlDesk Next Generation Migration of ControlDesk 3.x Automation*. 

---

- sdmlib
- officelibs (word, excel, powerpoint)

The following new Python modules are available for some dSPACE releases. These are compatible with the old ones but provide additional features.

- rtplib2
- matlablib2
- rs232lib2

For further information, refer to *Test Automation Python Modules Reference*.

The sdmlib Python module must be replaced either by the stimulus feature of the HIL API MAPort implementation or by Real-Time Testing scripts. Stimulus files in the new STZ format can be created by using the Signal Editor of ControlDesk Next Generation or by converting existing files via the sdm2stz converter. For download and instructions, refer to http://www.dspace.com/go/sdm2stz.

The cdautomationlib Python module must be replaced by other products provided by dSPACE, for example, the automation interface of ControlDesk Next Generation or the dSPACE HIL API Python implementation, refer to *ControlDesk Next Generation API Reference* and *dSPACE HIL API Python Implementation Document*.

The Python modules wordlib and excellib are provided as support downloads for compatibility reasons at http://www.dspace.com/go/dSPACEOfficeLibs.
## General Information on Using Python Installations

**Objective**  
The following information is relevant if you want to use both Python versions on your computer.

<table>
<thead>
<tr>
<th>Using Python 2.5 and Python 2.7 in parallel</th>
<th>Both Python versions can be used in parallel on your computer with the following restrictions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- The file associations for PY and PYW files can only be set to one Python version. This is usually the latest installed Python version.</td>
</tr>
<tr>
<td></td>
<td>- Environment variables are used by both Python versions. Their values, for example, for PYTHONHOME, must be set to the Python installation you want to work with. For an overview of the environment variables set by Python, refer to <a href="http://docs.python.org/2/using/cmdline.html">http://docs.python.org/2/using/cmdline.html</a>.</td>
</tr>
</tbody>
</table>

| Using dSPACE test automation with both Python versions in parallel | If your test automation scripts use dSPACE Python modules, distributed either via the dSPACE Python 2.5 setup or via the dSPACE Python Extensions setup available up to dSPACE Release 2013-A, and you do not want to migrate your scripts, you have to work with both Python versions. |
Required Modifications in Your Scripts

Objective
Gives you information on modifications which you have to make in your scripts.

Where to go from here
Information in this section

<table>
<thead>
<tr>
<th>Migrating Changes in the Python Core Package</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrating Changes in the PyWin32 Package</td>
<td>15</td>
</tr>
<tr>
<td>Using the dSPACE Library</td>
<td>16</td>
</tr>
<tr>
<td>Enhancements to the Standard Python 2.7</td>
<td>17</td>
</tr>
<tr>
<td>Distribution</td>
<td></td>
</tr>
</tbody>
</table>

Migrating Changes in the Python Core Package

Objective
When migrating your Python 2.5 applications to Python 2.7, note the following changes in the Python Core package.

Information in this topic

- Incompatible byte code on page 12
- Attribute from HierList class renamed on page 12
- Stricter evaluation of hexadecimal values on page 13
- Stricter check on argument list for listdir method on page 13
- Changed standard encoding on page 13
- Changed string representation of time structure on page 14

Incompatible byte code
The byte code of PYC files has changed. The interpreter of Python 2.7 does not start PYC files generated with Python 2.5.

- Recompile the PY files with Python 2.7.

Attribute from HierList class renamed
The list attribute of the HierList(object.object) class has been renamed to listControl. If the attribute is explicitly used in your source code or addressed via self._dict_, an exception is thrown because the attribute cannot be found.

- Change the list entries in your source code to listControl.
- or
- Implement a case construction to switch between the attribute names.
The `HierList` class is defined in `\Lib\site-packages\pythonwin\pywin\tools\hierlist.py`

**Stricter evaluation of hexadecimal values**

With Python 2.7, it is not sufficient to specify 0 as 0x.
- Use 0x0 instead of 0x to specify 0 in hexadecimal format.

To get the hexadecimal value of a string, you have to convert it to an integer value, for example, `int(HexStringValue, 16)`.

**Stricter check on argument list for listdir method**

With Python 2.7, the `os.listdir(path)` method does not accept an empty string as an argument.
- Use `os.listdir('')` instead of `os.listdir('')`.

**Changed standard encoding**

With Python 2.7, the standard encoding is set to ASCII independently of the specified system encoding. This might lead to the following problems.

**Locale default string encoding**

If a script is parsed in Python 2.7 and no script encoding is declared, the Python standard default encoding, ASCII, is applied. If the script contains characters which are not ASCII characters, script parsing fails.

You can check the current setting with the following script:

```python
import locale
system_encoding = locale.getdefaultlocale()[1]
```

- You should always define script encoding to clarify script usage. The encoding has to be declared in the first or second line of the script.
  The following example declares the latin-1 encoding.
  ```
  # -*- coding: latin-1 -*-
  #
  ```
  or

- You can set the encoding globally by editing the `setencoding` function in `\Lib\site.py`.

```python
if 0:
    # Enable to support locale aware default string encodings.
    import locale
    loc = locale.getdefaultlocale()
    if loc[1]:
        encoding = loc[1]
```

Replace `if 0:` by `if 1:`.
For further information on declaring encodings, refer to 2.1.4 Encoding declarations in the The Python Language Reference in the Python 2.7 documentation.

String conversion  If you encode a unicode string with a string conversion, for example, `str(u'ä')`, you get a `UnicodeEncodeError` exception when using it with Python 2.7.

Use the `encode` method to cast a unicode string to a string, for example, `u'ä'.encode('cp1252')`.

For further information on using unicode strings, refer to http://docs.python.org/2/howto/unicode.html.

---

### Changed string representation of time structure

With Python 2.7, the string representation of the time structure returns a more descriptive string than with Python 2.5. Parsing the returned string will now fail.

Example:
```
repr(time.localtime())
```

Python 2.5 returns: `'(2013,7,8,7,46,15,0,189,1)'`

Python 2.7 returns: `'time.struct_time(tm_year=2013, tm_mon=7, tm_mday=8, tm_hour=7, tm_min=46, tm_sec=5, tm_wday=0, tm_yday=189, tm_isdst=1)'`

Add a tuple conversion to get the same return value.
```
repr(tuple(time.localtime()))
```
Migrating Changes in the PyWin32 Package

Objective
When migrating your Python 2.5 applications to Python 2.7, note the following changes in the PyWin32 package.

Information in this topic

Removed manifest reference to MSVCR90
With Python 2.7, pythoncom27.dll does not contain a manifest reference to MSVCR90.dll. If your script tries to access MSVCR90.dll, an exception is thrown. The calling application is now responsible for making MSVCR90.dll available.

- Copy MSVCR90.dll to the folder the calling application is stored in.

SAFEARRAY discontinued
Calling the pythoncom.SAFEARRAY class leads to an error. This class is no longer supported.

- Use the new VARIANT class.

   Example:
   # Python 2.5
   pythoncom.SAFEARRAY([1,2,3,4,5],1,pythoncom.VT_R8)

   # Python 2.7
   from win32com.client import VARIANT
   VARIANT(pythoncom.VT_ARRAY | pythoncom.VT_R8,[1,2,3,4,5])

Bug in the Format method of pywintypes.Time
The Format method of a pywintypes.Time object fails if the date returned by another COM method is earlier than 1900.

- To format the elements of the date (year, month, day), access them separately.

Bug in the os.spawnv and os.spawnve functions
The os.spawnv and os.spawnve functions now do not accept an empty list for the process arguments.

- Provide at least one element in the list of arguments.
Using the dSPACE Library

Objective

The Python 2.5 installation provided by dSPACE was already adapted to dSPACE-specific requirements. Some classes and methods are therefore not available in the Python 2.7 standard distribution. They are now provided by a dSPACE library that comes with the dSPACE software products that use Python. You only have to add an import command in your scripts to access this library.

Information in this topic

Using the Enums class on page 16
Using the GetComIdentity method on page 16
Using the IsTypeOf method on page 16

Using the Enums class

The `win32com.Enums` class is not available in standard Python. It is provided by the dSPACE library.

- Add the following command to your scripts that use the `Enums` class:
  ```python
  from dspace.com import Enums
  ```

Using the GetComIdentity method

The `GetComIdentity` method in `win32com.client` is not available in standard Python. It is provided by the dSPACE library.

- Add the following command to your scripts that use the `GetComIdentity` method:
  ```python
  from dspace.com import GetComIdentity
  ```

Using the IsTypeOf method

The `IsTypeOf` method in `win32com.client` is not available in standard Python. It is provided by the dSPACE library.

- Add the following command to your scripts that use the `IsTypeOf` method:
  ```python
  from dspace.com import IsTypeOf
  ```
## Enhancements to the Standard Python 2.7 Distribution

### Objective

There are some dSPACE-specific enhancements to the standard Python 2.7. These either ensure the same behavior as before or solve known bugs.

### Enhancements to get the same behavior as with Python 2.5

The following changes have been made to ensure that the Python installation behaves identically to the Python 2.5 dSPACE installation:

- 64-bit signed and unsigned integer data types can now be used in an automation call.
- Integer data types are now converted to valid ranges. Formerly, they were simply mapped to the double data type.
- Enhanced way to obtain error information after a failed automation call.
- The signed char data type is now handled as an array instead of converting it to an unsigned char data type.
- A PyOleNothing data type has been added to support interfaces which make use of the Visual Basic Nothing object.
- The Python Imaging Library (PIL) has been enhanced by a better dependency management to correct the wrongly referenced C-Runtime.
- By default, PY and PYW files are associated with Python.exe, which you can double-click to execute a file.
  
  The default behavior of Python 2.7 has been changed by dSPACE. The files are now opened in PythonWin.

### Enhancements to solve known Python bugs

The following changes have been made to solve known bugs from Python 2.7:

- If an Item method was marked as the default property, the implementation for the index operator was not generated.
- The memory was not correctly freed if strings or interfaces were handled in arrays.
- The deadlock situation in multi-threaded scenarios using pythoncom has been solved.
- The default path for generated COM wrappers was not allowed to be located in a read-only location.
- The conversion of strings to CLSIDs which are created if a Progid is given to the function was erroneous.
- The win32 API method FormatMessage that uses a lock internally lead to a multi-threading deadlock.
- Python crashed if a COM object implemented in Python was released after the Python interpreter was shut down.
- Python crashed if the Python windows hook for windows messages was still installed after the interpreter was shut down.
- The generation of wrappers for standard COM interfaces was disabled if they were referenced in a type library.
- Python still held a reference to an enumerator although it was reset in a COM server.

For the latest information on bugs in Python 2.7 and their solutions, see http://bugs.python.org.

All changes to the PyWin32 package are reported in the bug tracking system, for example:

- https://sourceforge.net/p/pywin32/patches/123/
- https://sourceforge.net/p/pywin32/patches/127/
- https://sourceforge.net/p/pywin32/patches/128/
- https://sourceforge.net/p/pywin32/patches/130/
- https://sourceforge.net/p/pywin32/patches/123/
- https://sourceforge.net/p/pywin32/patches/127/
- https://sourceforge.net/p/pywin32/patches/128/
- https://sourceforge.net/p/pywin32/patches/130/
- https://sourceforge.net/p/pywin32/bugs/609/
- https://sourceforge.net/p/pywin32/bugs/630/
- https://sourceforge.net/p/pywin32/bugs/638/
- https://sourceforge.net/p/pywin32/bugs/639/
- https://sourceforge.net/p/pywin32/bugs/640/
- https://sourceforge.net/p/pywin32/bugs/643/
- https://sourceforge.net/p/pywin32/bugs/649/
- https://sourceforge.net/p/pywin32/bugs/650/
- https://sourceforge.net/p/pywin32/bugs/651/
- https://sourceforge.net/p/pywin32/bugs/652/
- https://sourceforge.net/p/pywin32/bugs/653/

To identify the PyWin32 files changed by dSPACE the version number is changed from 218.0 to 218.10.