PyWBEM Python WBEM Client: Overview

Karl Schopmeyer
k.schopmeyer@opengroup.org
Andreas Maier
MAIERA@de.ibm.com

February 2016

SNIA SMI plugfest #2

Version: 0.6_A – 17 Feb
Version 0.7_ks - 20 Feb.
Version 0.8_ks – 22 Feb
Version 0.9_ks 25 Feb

PyWBEM Client: Overview

- Pure Python code
  - Python 2.6, 2.7 (WIP: 3.4, 3.5)
- Supports DMTF CIM-XML protocol
  - Client library with a pythonic API
  - Indication listener (experimental)
- Utilities:
  - MOF compiler
  - Command line interface utility
- LGPL 2.1 license

Availability

- Client package "pywbem" available in Pypi repository
- Client package available on some Linux distributions
  - Ex. Ubuntu as python-pywbem (v. 0.7.0)
- Directly available from pywbem project on Github:
  - pywbem is a github group with 4 code repositories (pywbem, cimserver, yawn, pyprov) and a doc repository (pywbem.github.io)
- Download links on PyWBEM github web site:

  `http://pywbem.github.io`

- This presentation concentrates on:
  - **pywbem** – PyWBEM Client project (Python client and related utilities)
  - **pywbem.github.io** – Documentation for the PyWBEM projects

Possible usage

- WBEM Client infrastructure for python based products
- Client infrastructure for test tools
  - Multi-platform because python multiplatform
  - Simple to write tests in python
- Basis for extended WBEM client tools
  - WBEM/CIM browsers
  - MOF viewers
  - Etc.
- Pywbem is equivalent to client side infrastructure support in most platforms (OpenPegasus, etc.)

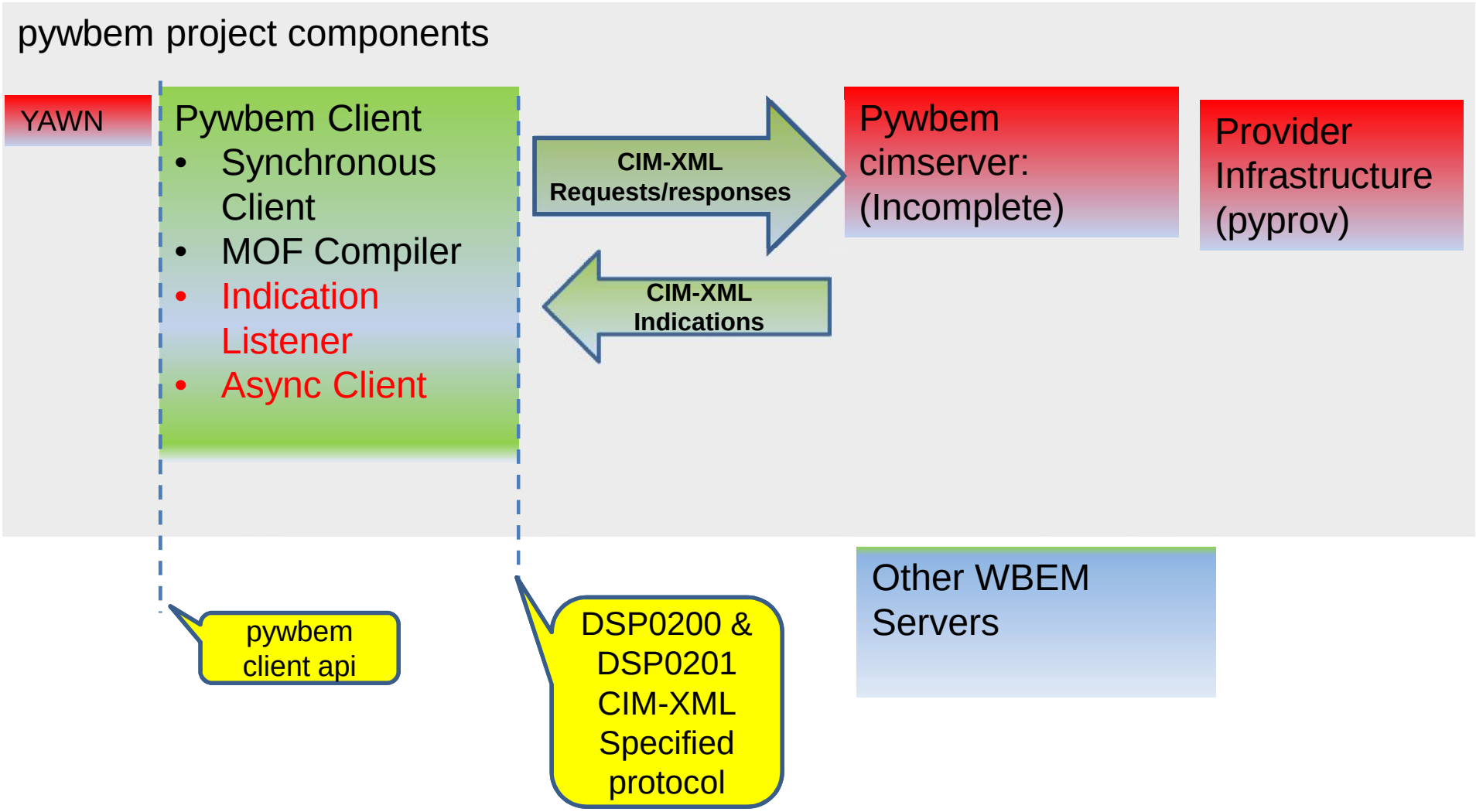
Project History

- Originated about 2005 by HP
- Available originally on SourceForge 2005 (v0.3)
- Extended and grown by HP and Novell on SourceForge
- Development abated about 2009 with version 0.7.0
- Renewed interest by IBM in 2013 and activity restarted
- 2014 early release candidate for 0.8.0 released on source forge (rc1)
- 2015 moved to Github
- Working now to release final 0.8.0 release

PyWBEM Client Overview

- Implements DMTF specifications DSP0200 and DSP0201 (CIM-XML client protocol)
- Implements all client operations (DSP0200 1.2/1.3)
- Missing today
  - Pull Operations defined in DMTF DSP0200 1.4
- Experimental Today
  - CIMlistener (irecv directory)

Pywbem Overview



Overall Status Today

- Version 0.7.0
  - Released 2008 on SourceForge and Pypi by Novell
  - Supports Python 2.6, 2.7
  - Limited tests
  - Distributed on multiple linux platforms
  - Integrated client, compiler, and other experimental components (cimserver, provider, etc. into single repository.
  - See Change log for more detailed information

Overall Status (cont)

- PyWBEM Client version 0.8.0
  - Plan
    - In process for over 2 years and several release candidates
      - rc1 – rc3
    - Concentration on PyWBEM Client
    - Release of final 0.8.0 in near future
- rc3
  - Many changes since 0.7.0, see NEWS file for details
- Rc4 – internal may not release rc4
  - In testing today
  - Final 0.8.0 release to be based on rc4

0.7.0 – 0.8.0 changes

- Significant code cleanup
- Major documentation update
- Major extensions to test environment
- Support for Python 3 (work in progress)
- Break out code into separate repositories
- Add web documentation and separate doc repository
- SSL/Crypto library cleanup
- Move source from SourceForge to Github
- See NEWS file for more details

Expected 0.8.0 Release

- Q1 2016
- Activity today to Complete 0.8.0 release
  - Fix bugs in github issues list

Post 0.8.0 Work

- Add Pull Operations
- Add more tests
- Review other code and determine direction
  - Twisted client
  - Listener (irecv)
- Python Providers
- Performance improvements
- Add anything necessary for complete DMTF compliance

Installation

- Latest release from Pypi (currently 0.7.0)
  - pip install pywbem
- Latest dev. code from Github (0.8.0 dev)
  - git clone git@github.com:pywbem/pywbem.git
  - cd pywbem
  - python setup.py install
- 0.8.0 rc3 release candidate:

  http://pywbem.github.io/pywbem/installation.html

Technical Overview

Getting started with the client

- Create a connection
- Execute WBEM client operations
- Disconnect connection
- Error handling through exceptions with standard CIM Error status and objects

CIM Objects Supported

- CIMClass
- CIMInstance
- CIMInstanceName
- CIMQualifierType
- CIMProperty
- CIMMethod
- CIMParameter
- CIM Primitive Data Types
  - Supports all of the CIM primitive types

CIMInstance

- Pywbem class (CIMInstance)
- Attributes of class
  - classname: string
  - properties: dictionary (optional)
  - qualifiers: dictionary of qualifier values(optional)
  - path: Instance of CIMInstanceName object
  - Property_list
- Methods
  - copy, update, get, tocimxml, tomof …
- Examples

```
filter = pywbem.CIMInstance('CIM_IndicationFilter',
                            {'Name': 'pywbem_test',
                             'Query': 'SELECT * FROM CIM_Indication',
                             'QueryLanguage': 'WQL'})
```

CIM Data Types

- Supports all CIM data types:
  - Uint8/Sint8
  - Uint16/Sint16
  - Uint32/Sint32
  - Uint64/Sint64
  - String
  - Real32/Real64
  - DateTime (separate class)
  - Embedded Instance
  - TODO fix per http://pywbem.github.io/pywbem/doc/0.8.0/doc/pywbem.cim_types-module.html

Client request operations

- pywbem supports all of the interfaces defined in DSP0200 (WBEM Operations over CIM-XML, v 1.2).
- Pull Operations (CIM/XML DSP0200 v 1.4) (Planned for pywbem 0.9.0)

Client Request Operations

- Instance
  - GetInstance
  - CreateInstance
  - ModifyInstance
  - DeleteInstance
  - EnumerateInstances
  - EnumerateInstanceNames
  - Associators
  - References
  - AssociatorNames
  - ReferenceNames
  - InvokeMethod
  - ExecQuery

- Class/Qualifier
  - GetClass
  - CreateClass
  - ModifyClass
  - DeleteClass
  - EnumerateClasses
  - EnumerateClassNames
  - GetQualifier
  - SetQualifier
  - DeleteQualifier
  - EnumerateQualifiers

NOTE: Qualifier operations operate on qualifier types (i.e. qualifier declarations)

Simple example: EnumeratInstances

```
#   Simple client gets instances from a class and displays them
import pywbem
username = <name for user>
password = <password for user>
servername = localhost
classname = ‘CIM_OperatingSystem’

client = pywbem.WBEMConnection(
    'https://%s' % servername, (username, password))

instances = client.EnumerateInstances(classname)
if len(instances) == 0
    exit
print instances[0].items()
[(u'Parameters', [u'init']),
 (u'CSName', u'nautilus3.asiapacific.cpqcorp.net'),
 (u'RealUserID', 0L),
 ...
# display one instance in mof format
print(mof {}).format(instance[0].tomof()))
… returns formatted mof representation of the object
```

Simple example: GetInstance

```
# Simple client gets instances from a class and displays them
import pywbem
server_name = localhost
Instance_name = ‘CIM_ComputerSystem’ TODO build instance name

# try block to cover both connection and request
try:
    client = pywbem.WBEMConnection('https://%s' % server_name)

    instance = client.GetInstance(instance_name)
    # display instance in mof format
    print(‘mof: {}’).format(instance.tomof()))
    print(‘xml: {}’.format(instance Todo)

except pywbem.CIMError, arg:
    print 'CreateInstance: %s' % arg[1]
    sys.exit(1)
```

WBEM Server compatibility

- Compatible with any server that supports DMTF CIM-XML protocol (DSP0200/DSP0201)
  - OpenPegasus
  - SFCB
  - WBEM Solutions Java WBEM Server
  - EMC
  - Others

wbemcli – A simple Python CLI

- Python command line tool to execute client methods interactively within Python environment
- Connects to WBEM server when initiated and then returns to Python interactive interpreter
- Includes functions for execution of client request operation:
- Usage
  - `wbemcli.py HOST [-u USER] [-p PASS] [-n NAMESPACE] [-p PORT] [--no-ssl]`

wbemcli (cont)

- Saves command history
- Pretty print of CIM objects
- Access to all of Python interpreter

PyWBEM MOF Compiler

- Compile CIM classes, qualifier types, instances from MOF files
- Allows creating classes, instances, qualifier types.
- Inserts created objects into a repository
- Repository API is pluggable:
  - Repository implementation can be provided by the user
  - Default is whatever is open as wbem connection

Other subjects not discussed here

- CIM Listener (pywbem repository)
  - Code exists but we do not know status
  - Need to add tests to confirm status
- Twisted client (pywbem repository)
  - Async client based on twisted library
  - Status unkonwn
- pyprov (Separate repository)
  - Separate repository
  - There were several project to build python providers
  - Status of code unknown to current team
- cimserver (Separate repository)
  - Experimental and incomplete
  - Not sure what will happen to this in future
- yawn (Extension for cim browsing through web server)

More Information

- See PyWBEM Client documentation online at:
  - http://pywbem.github.io/pywbem/
- Includes info on:
  - Installation
  - API documentation
  - Usage Tutorial
- Engage with PyWBEM community, for:
  - reporting issues (github issues)
  - asking for feature requests (github issues)
  - Contributing (for example from github fork)