



December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA

# WS-Management Application enablement

Klaus Kämpf



**Novell**®

# Overview

- Motivation
- Openwsman Server
- Brute force enablement - YaST
- Client side
- Following the semantics – Hal
- Demo

# Why application enablement ?

- Remote access
- Standardized protocol
- Interoperability
- WebService architecture
- Because you can

# Why through WS- Management ?

- Single protocol
- Single daemon
- Single open port
- Security
- Firewall friendly
- Easy to code

# Openwsman Server Side

## HTTP Server

### openwsmand

Listen

Authorize

Dispatch

### Endpoints

Invoke  
Identify

WS-Transfer  
Create  
Delete  
Get  
Put

WS-Enumeration  
Enumerate  
Pull  
Release

WS-Eventing  
Subscribe  
Unsubscribe  
Renew

- Resource definition

```
1  #include <wsman-declarations.h>
2
3  struct __my_resource
4  {
5      char *data;
6  };
7  typedef struct __my_resource MyResource;
8
9  SER_DECLARE_TYPE(MyResource);
10 DECLARE_EP_ARRAY(MyResource);
11
12 #define XML_NS_MY "http://schema.wsman.com/Resource/my/1-0"
```

- Plugin Registration

```
1  #include "my-resource.h"
2
3  SER_START_ITEMS( MyResource )
4  ...
5  START_END_POINTS( MyResource )
6      END_POINT_TRANSFER_DIRECT_CREATE( MyResource, XML_NS_MY ),
7      END_POINT_TRANSFER_DIRECT_GET( MyResource, XML_NS_MY ),
8      ...
9  START_NAMESPACES( YaST )
10     ADD_NAMESPACE( XML_NS_MY, "My" ),
11     ...
12
13 void get_endpoints( void *self, void **data ) { ... }
14 int init( void *self, void **data ) { ... }
15 void cleanup( void *self, void *data ) { ... }
16 void set_config( void *self, dictionary *config ) { ... }
```

# Planning the enablement

- Honor WS-Man semantics
- Namespace
- Endpoints (Create, Get, ..., Invoke)
- Define the goal
- Type of Resource
- Testing

# Client side tools

- winrm (Windows)
- wsmcli (Openwsman)
- C / C++ Code
- Scripting languages (Python, Ruby)

# Example: openwsman-yast

- Namespace

`http://schema.opensuse.org/YaST/wsman-schema/10-3`

- Resource Type

`char *ycp;`

- Resource Name

`"YCP"`

- Endpoints

`END_POINT_CUSTOM_METHOD()`

- Method Name

`"eval"`

# Example: openwsman-yast

- Windows Scripting

```
winrm invoke eval http://schema.opensuse.org/YaST/wsman-schema/10-3/YCP  
-file:ycp.xml -username:wsman -password:secret -r:1.2.3.4:8889/wsman  
-auth:basic
```

# Example: openwsman-yast

- Windows Scripting

```
winrm invoke eval http://schema.opensuse.org/YaST/wsman-schema/10-3/YCP  
-file:ycp.xml -username:wsman -password:secret -r:1.2.3.4:8889/wsman  
-auth:basic
```

```
<p:eval_INPUT xmlns:p="http://schema.opensuse.org/YaST/wsman-schema/10-3/YCP">  
  <p:ycp>{ return "Hello, world!"; }</p:ycp>  
</p:eval_INPUT>
```

# Example: openwsman-yast

- xml output
  - Pass structured data (Array, Hash, ...)
- Ruby parser
  - Convert to objects of target language
- Package available for OpenSUSE 10.3
  - openwsman-yast-1.2.0

# HAL

- Hardware Abstraction Layer
- Inventory
- Resources
- Semantics

# Example: openwsman-hal

- Implements Create, Get, Put, Delete, Enumerate endpoints
- Example: Enumerating sound devices

# Example: openwsman-hal

- Implements Create, Get, Put, Delete, Enumerate endpoints
- Example: Enumerating sound devices

```
1  ns = "http://schema.freedesktop.org/HAL/wsman-schema/0-8-5"  
2  uri = ns + "/org/freedesktop/Hal/devices"  
3  
4  options.selector_add( "capability", "alsa" )  
5  result = client.enumerate( uri, options )
```

# WS-Management Application Enablement

Screenshot

# WS-Management Application Enablement

- Screenshot
- Full demo in Interop Lab
- Mail: [kkaempf@suse.de](mailto:kkaempf@suse.de)
- <http://www.openwsman.org>
- <http://www.opensuse.org>
- <http://build.opensuse.org>

# That's all folks !

Thank you !