

## **1 Target consumers of this standard**

The target consumer of this standard is a desktop application. Not all desktop services nor a Internet or enterprise service but a desktop application.

Desktop applications like browsers, office suites, E-mail clients and instant messengers if they store personal configuration data sometimes also called the preferences of the user.

This standard is about personal preferences that influence the behaviour of desktop applications.

## **2 Technology dependencies of this standard**

For a desktop developer, the only dependency for using this standard is the D-BUS standard and the service implementation that will implement it.

This standard doesn't add any other dependency. An implementation that respects this standard, can be used by any consumer of this standard.

## **3 Serve and consume using D-BUS**

This standard defines a client versus service protocol. The service delivers the information and that performs requests coming from the client.

The client consumes the service. D-BUS is the transport layer technology between client and service.

The desktop application is the client. The implementation of this standard is the service.

## 4 Schemas

This standard defines the schema DTD

Each schemas file may describe multiple keys, the information for each key includes:

- The root of the keys in the schemas file
- The type for each the value of the key in D-BUS type signature format.
- Localized short and long descriptions of the settings
- A default value for the settings.

TODO

## 5 The protocol

### 5.1 The errors

NOSUCHKEYERROR	"No such key error"
KEYISREADONLYERROR	"Key is readonly error"
UNKNOWNERROR	"Unknown error"

### 5.2 The types

#### struct STRUCT

STRING	the member-name
value STRUCT	the value

#### The value STRUCT

UINT32	the type
VARIANT	the data

#### data VARIANT

can be:

STRING	the string data	or
INT64	the integer data	or
BOOLEAN	the boolean data	or
DOUBLE	the double data	or
ARRAY of value STRUCT	the list data	or
ARRAY of struct STRUCT	the struct data	

#### The METAINFO DICT

STRING	name of the information
VARIANT	the information

#### metainfo VARIANT

can be:

STRING	the string data	i.e. a description
INT64	the integer data	i.e. a UNIX timestamp

### 5.3 Getting values

Gets the value of a key. In case no value is available, the default from the schema of the key is returned.

In case there's also no schema an NOSUCHKEYERROR error is returned.

**org.freedesktop.configuration.GetValues**

**org.freedesktop.configuration.GetValue**

As methods:

ARRAY of value STRUCT	GetValues	(STRING root)
value STRUCT	GetValue	(STRING key)

### 5.4 Setting values

This action sets (overwrites or creates) the value of a key.

In case the key is read-only, a KEYISREADONLYERROR error is returned.

SetValue	(STRING key, value STRUCT value)
----------	----------------------------------

### 5.5 Getting metainfo

Gets the metainfo of a key. In case no metainfo is available, the default from the schema of the key is returned.

In case there's also no schema an NOSUCHKEYERROR error is returned.

**org.freedesktop.configuration.GetMetaInfo**

As a method:

METAINFO STRUCT	GetMetaInfo	(STRING key)
-----------------	-------------	--------------

### 5.6 Setting metainfo

This action sets (overwrites or creates) the metainfo of a key.

In case the key is read-only, a KEYISREADONLYERROR error is returned.

**org.freedesktop.configuration.SetMetaInfo**

As a method:

SetMetaInfo	(STRING key, METAINFO STRUCT metainfo)
-------------	--

## 5.7 Subscribing to notification of changes

A client will only receive a `KeysChanged` event about a specific key if it's subscribed to events about that key. These actions subscribe and unsubscribe a client.

TODO: I don't know how to model/use dbus signal match rules yet.

## 5.8 Receiving notification of changes

This signal happens (on the client) if the client is subscribed to a key that was changed.

### **org.freedesktop.configuration.KeysChanged**

This is a signal:

```
KeyChanged          (STRING key, VARIANT newvalueorremoved, UINT32 next)
```

## 5.9 Removing

This action removes all keys starting from the root.

### **org.freedesktop.configuration.RemoveKeys**

As a method:

```
RemoveKeys          (STRING root)
```