

# Table of Contents

<b>How to Manage a Set of Services.....</b>	<b>1</b>
Introduction.....	1
Creating a Service Set.....	1
Activating a Service Set.....	1
Inheriting between Service Sets.....	2
Sharing Service Sets between Threads.....	2
Review Status.....	3

# How to Manage a Set of Services

Complete:

## Introduction

Multiple 'sets of services' can be used within an application, although only one set can be active per thread at an instance in time.

## Creating a Service Set

Only the application 'main' routine and an EventProcessor are allowed to create new 'service sets'. There are several interfaces to create a new service set: via a parameter set or by inserting an already created service instance.

A new service set can be created by passing an `std::vector<edm::ParameterSet>` to `edm::ServiceRegistry::createSet()`. The method returns an `edm::ServiceToken` which is later used to make a service set active. The `edm::ParameterSet` used to describe a Service expects a Parameter named '@service\_type' of type 'string' to hold the name of the Service type that is to be created. The other Parameters used to configure the service should also be set in the `edm::ParameterSet`. [NOTE: in CMSSW\_0\_1\_0 the name was 'service\_type' instead of '@service\_type']

```
std::vector<edm::ParameterSet> params;
edm::ParameterSet pSet;
pSet.AddParameter("@service_type", std::string("Tracer"));
pSet.AddParameter("indentation", std::string("++"));
params.push_back(pSet);

edm::ServiceToken token = edm::ServiceRegistry::createSet(params);
```

One can also create a service set by first creating an instance of the service and then passing it to `edm::ServiceRegistry::createContaining()`. The method returns an `edm::ServiceToken` which is later used to make a service set active. One can either pass an `std::auto_ptr<T>` or a `boost::shared_ptr< edm::serviceregistry::ServiceWrapper<T> >` to the `createContaining()` method, where T is the actual C++ type of the service. The choice depends on if you do not need to change the service later (then use `std::auto_ptr`) or if you must allow for changes (then use `boost::shared_ptr`). [NOTE: this option was added after CMSSW\_0\_1\_0]

```
std::auto_ptr<Foo> foo( new Foo() );

edm::ServiceToken token = edm::ServiceRegistry::createContaining(foo);
```

## Activating a Service Set

A service set is made active (that is, the services it provides can be accessed via `Service<>`) through the use of `edm::ServiceRegistry::Operate`. The `Operate` class is meant to live on the stack and is passed an `edm::ServiceToken` in its constructor. As part of the construction, `Operate` will make all the services described by `edm::ServiceToken` active. When the `Operate` goes out of scope, its destructor will replace the present 'service set' with the set that was 'active' before the `Operate` was constructed.

```
edm::ServiceRegistry::Operate( token );
...
```

The use of `edm::ServiceRegistry::Operate` allows services sets to be nested where what services are available at an instant in time is dependent on how deep you are in the call stack.

## Inheriting between Service Sets

It is possible to share all or some of the services between two service sets. This is done by passing a previously made `edm::ServiceToken` to `edm::ServiceRegistry::createSet()` along with the `std::vector<edm::ParameterSet>` used to configure the additional services and a `edm::serviceregistry::ServiceToken` enumeration value which states how overlaps between the `edm::ServiceToken` and the `std::vector<edm::ParameterSet>` should be handled. The results of the call to `createSet()` will be an `edm::ServiceToken` that can contain Services 'inherited' from the other `edm::ServiceToken`.

```
std::vector<edm::ParameterSet> params;
edm::ParameterSet pSet;
pSet.AddParameter("@service_type", std::string("Tracer"));
params.push_back(pSet);
```

```
edm::ServiceToken childToken = edm::ServiceRegistry::createSet(parentToken, kOverlapIsError, para
```

`edm::CMS.EventProcessor` can also inherit services by calling the forms of its constructor that take an `edm::ServiceToken` and `edm::serviceregistry::ServiceLegacy`.

Allowed values of `edm::serviceregistry::ServiceLegacy` are as follows

name	description
<code>kOverlapIsError</code>	If an overlap occurs, then an exception will be thrown
<code>kTokenOverrides</code>	If an overlap occurs, the service described in the <code>edm::ServiceToken</code> is used
<code>kConfigurationOverrides</code>	If an overlap occurs, the service not in the <code>edm::ServiceToken</code> is used

## Sharing Service Sets between Threads

Each thread in the application will have its own `edm::ServiceRegistry`. However, a service set can be shared between threads. This is accomplished by passing an `edm::ServiceToken` between two threads (this operation is guaranteed to be safe).

If you want a child thread to inherit the presently active service set from the parent thread, then

- in the parent thread call `edm::ServiceRegistry::instance().presentToken()` to get an `edm::ServiceToken` for the presently active service set
- pass that `edm::ServiceToken` to the child thread
- in the child thread construct an `edm::ServiceRegistry::Operate` on the stack

```
struct PassServices {
    PassServices(edm::ServiceToken iT): token_(iT) {}
    void operator() () {
        //make the services available
        edm::ServiceRegistry::Operate(token_);
        ...
    }
    ...
    edm::ServiceToken token_;
};

...
//pass the present services to the new thread
```

```
PassServices passRun(ServiceRegistry::instance().presentToken());
boost::thread newThread(passRun);
```

NOTE: The ServiceRegistry system itself is thread safe, but it is up to the individual service developers to guarantee that their service is safe to operate in multiple threads.

## Review Status

Reviewer/Editor and Date (copy from screen)	Comments
ChrisDJones - 13 Sep 2005	page author
ChrisDJones - 05 Oct 2005	page content last edited
JennyWilliams - 07 Feb 2007	editing to include in SWGuide

Responsible: ChrisDJones

Last reviewed by: Reviewer

---

This topic: CMSPublic > SWGuideEDMServiceDocManaging

Topic revision: r6 - 06-Feb-2007 - 23:56:16 - JennyWilliams



Copyright &© by the contributing authors. All material on this collaboration platform is the property of the contributing authors.

Ideas, requests, problems regarding TWiki? Send feedback