# Registry Context

## 22-May-2000

**This version:**
Draft 0.2
**Technical Contact:**
Scott Hinkelman <srh@us.ibm.com>

**Copyright Statement:**
Copyright IBM Corp 2000 – Property of IBM

---

## Abstract

*Registries and Repositories have multiple aspects, such as content classification, content identification, actor roles, etc. This draft document addresses the distribution aspect of Registries, by defining a structure for context-based distributed Registries (a "RegistryContext") built on a hierarchical graph.*

*It is rooted in effective and proven design points and implementations from previous large-scale distributed system environments supported by IBM built on technology such as DCE, CORBA, and Java. It follows a fundamental principal of re-engineering earlier work in distributed object technology from the intra-enterprise space to the inter-enterprise space; in this case distributed Naming Services and Directory Services.*

## Status of this Document

This document has been specially prepared from work in progress in IBM research and development for consideration by the ebXML Registry and Repository work group.

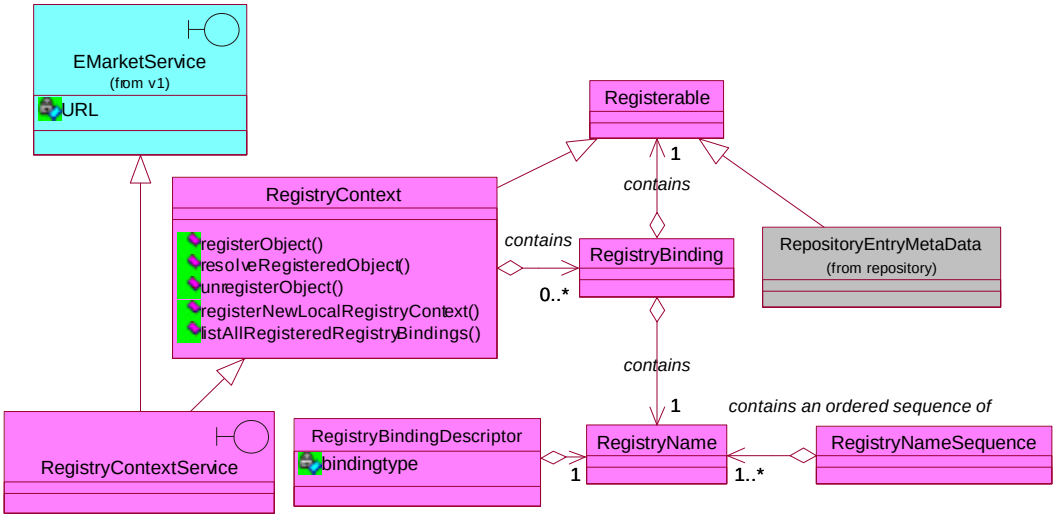
## Table of Contents

Copyright IBM Corp 2000 – Property of IBM

# 1 Introduction

This document defines a messaging interface concerned with the distribution topology of Registries based on hierarchical RegistryContexts. An approach based on RegistryContexts provides a model to support a specific distributed topic hierarchical Registries instantiations.

This document is based on registry context for a topic distribution mechanism, with assumption that hierarchical topics (local or distributed) are a natural preamble for classification schemes (with functionality such as classification-scheme-search, transformations, etc) of registered metadata of Repository content information. This implies that topic hierarchy should provide context for a registered RepositoryEntryMetaData with various classification schemes, and the functionality over those components such as search/query (not addressed in this document). A distributed context model is defined for which component classification schemes will be built on top of. Registered RepositoryEntryMetaData components are not discussed in this document.




A *RegistryName* to *RegistryObject* association is called a *RegistryBinding*. A RegistryBinding is always defined relative to a *RegistryContext*. A RegistryContext is an object that contains a set of RegistryBindings in which each RegistryName is unique. Different RegistryNames can be bound to a RegisteredObject in the same or different RegistryContexts at the same time.

A RegistryObject specialization (a RegistryContext or a RepositoryEntryMetaData), which is bound into a RegistryContext is unaware of the fact that it has been associated to a RegistryName via a RegistryBinding, and that the RegistryBinding may be bound into a RegistryContext (not navigable).

A RegistryName is used to identify the binding within the RegistryContext for which it may be bound. A *RegistryNameSequence* is an ordered set of RegistryNames that can be used to resolve a RegisteredObject from a given target RegistryContext.

RegistryContextService is RegistryContext boundary interface and is an EMarketService. For the extent of the model scope of this document, a URL is inherited and is used to facilitate distribution of RegistryContexts through URL addressing.

A RegistryBindingDescriptor describes a RegistryBinding by identifying the type of binding and the RegistryName. RegistryBindingDescriptors are returned on list messages on RegistryContexts.

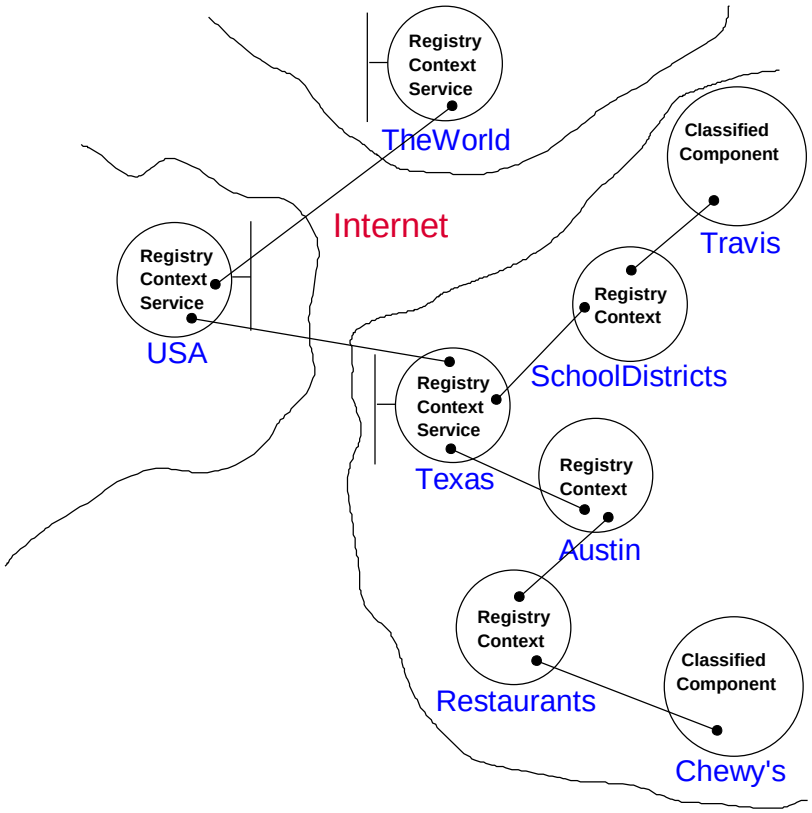
# 2 Distributed Registry Context Graph

RegistryContext objects are registered together through bindings to form a distributed *Registry Graph*.

Since RegistryContexts are RegistryObjects, they can be associated to a RegistryBinding, and then that binding be registered into another RegistryContext. If a RegistryContext is registered into a RegistryContext with a different URL than it's own URL, a distributed Registry Graph is formed.

The Registry Graph is a classic directed graph pattern composed of nodes and leafs, where the nodes are registrations of bindings for RegistryContexts, and the leafs are registrations of bindings for RepositoryEntryMetaData objects (which are also RegistryObjects).

*Resolving* a RegistryObject means to determine a registered RegistryObject associated with a specified RegistryNameSequence. A RegistryNameSequence contains one or more ordered RegistryNames used to define a path within a Registry Graph relative to a RegistryContext. A RegistryObject is always resolved or identified relative to a RegistryContext.




Each RegistryName within a RegistryNameSequence is used to identify a RegistryContext and the last RegistryName denotes the RegistryObject.

Copyright IBM Corp 2000 – Property of IBM

# 3 Message Interface Functional Specification

## 3.1 RegistryContext Interface

The RegistryContext Interface defines message operations for registration functionality with a Registry.

### 3.1.1 registerObject

| **void registerObject(registrybinding RegistryBinding, registrynamesequence RegistryNameSequence)**<br>Error message payloads: RegistryContextNotFound, InvalidRegistryNameSequence, AlreadyBound |
|---|

The registerObject message accepts a RegistryBinding (a RegistryName and a RegistryObject) and names a RegistryObject in a target RegistryContext.

If a RegistryContext is registered with this message, it will then participate, locally or through distributed traversal, in RegistryObject resolution when RegistryNameSequences with more than one RegistryName are passed for resolution on a resolveRegisteredObject message.

The return message payload is empty on success.

The return message payload contains a RegistryContextNotFound if the RegistryNameSequence parameter does not identify a RegistryContext.

The return message payload contains a InvalidRegistryNameSequence if the RegistryNameSequence parameter is invalid for the implementation.

The return message payload contains an AlreadyBound if the RegistryNameSequence parameter identifies an existing RegistryBinding.

### 3.1.2 resolveRegisterObject

| **RegisteredObject resolveRegisteredObject(registrynamesequence RegistryNameSequence)**<br>Error message payloads: RegistryContextNotFound, InvalidRegistryNameSequence |
|---|

The resolveRegisteredObject message retrieves a RegisteredObject. The invoked RegistryContext may cause distributed traversal to other RegistryContexts in order to resolve the specified RegistryNameSequence depending on the topology of the Registry Graph.

The return message payload contains a RegisteredObject on success.

The return message payload contains a RegistryContextNotFound if the RegistryNameSequence parameter does not identify a RegistryContext.

The return message payload contains a InvalidRegistryNameSequence if the RegistryNameSequence parameter has zero RegistryNames or is invalid for the implementation.

Copyright IBM Corp 2000 – Property of IBM

### 3.1.3 unregisterObject

| **void unregisterObject(registrynamesequence RegistryNameSequence)**<br>Error message payloads: RegistryContextNotFound, InvalidRegistryNameSequence |
|---|

The unregisterObject message unregisters a RegistryObject from a target RegistryContext. The invoked RegistryContext may cause distributed traversal to other RegistryContexts in order to resolve the specified RegistryNameSequence depending on the topology of the Registry Graph.

The return message payload is empty on success.

The return message payload contains a RegistryContextNotFound if the RegistryNameSequence parameter does not identify a RegistryBinding.

The return message payload contains a InvalidRegistryNameSequence if the RegistryNameSequence parameter is invalid for the implementation.

### 3.1.4 registerNewLocalRegistryContext

| **void registerNewLocalRegistryContext(registrynamesequence RegistryNameSequence)**<br>Error message payloads: RegistryContextNotFound, InvalidRegistryNameSequence, AlreadyBound |
|---|

The registerNewLocalRegistryContext message creates and registers a new RegistryContext within the URL of the invoked RegistryContext based on a specified RegistryNameSequence.

The return message payload is empty on success.

The return message payload contains a RegistryContextNotFound if the RegistryNameSequence parameter does not identify a RegistryContext.

The return message payload contains a InvalidRegistryNameSequence if the RegistryNameSequence parameter is invalid for the implementation.

The return message payload contains an AlreadyBound if the RegistryNameSequence parameter identifies an existing RegistryBinding.

### 3.1.5 listAllRegisteredRegistryBindings

| **RegistryBindingDescriptor[] listAllRegisteredRegistryBindings(registrynamesequence RegistryNameSequence)**<br>Error message payloads: RegistryContextNotFound, InvalidRegistryNameSequence |
|---|

The listAllRegisteredRegistryBindings message returns a list of all RegistryBindingDescriptors that describe the bound RegistryNames and the type of binding: either RegistryContext, or RepositoryEntryMetaData.

The return message payload contains an array of RegistryBindingDescriptors on success.

The return message payload contains a RegistryContextNotFound if the RegistryNameSequence parameter does not identify a RegistryContext.

The return message payload contains a InvalidRegistryNameSequence if the RegistryNameSequence parameter is invalid for the implementation.

# 4 References

[1] *OMG Naming Service Specification CORBA Services*, Object Management Group March 1995

[2] *Java Naming and Directory Interface Specification*, Sun Microsystems January 1998