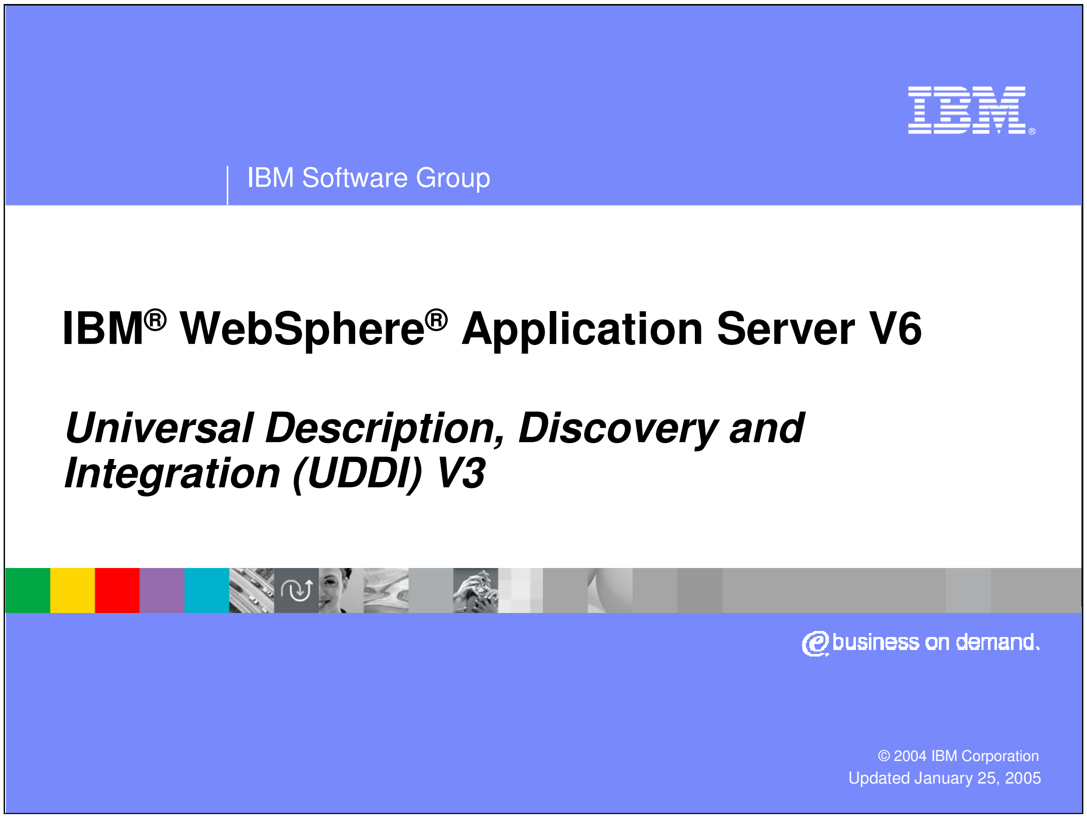


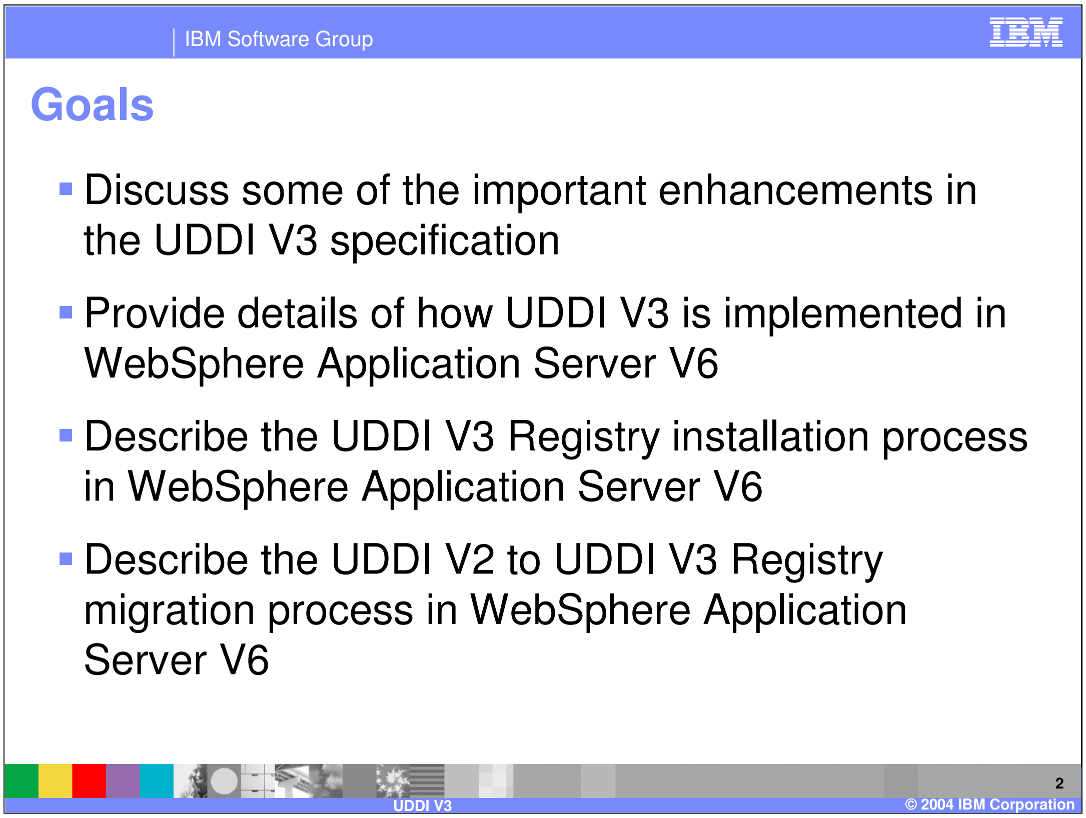
IBM Software Group

IBM® WebSphere® Application Server V6

Universal Description, Discovery and Integration (UDDI) V3

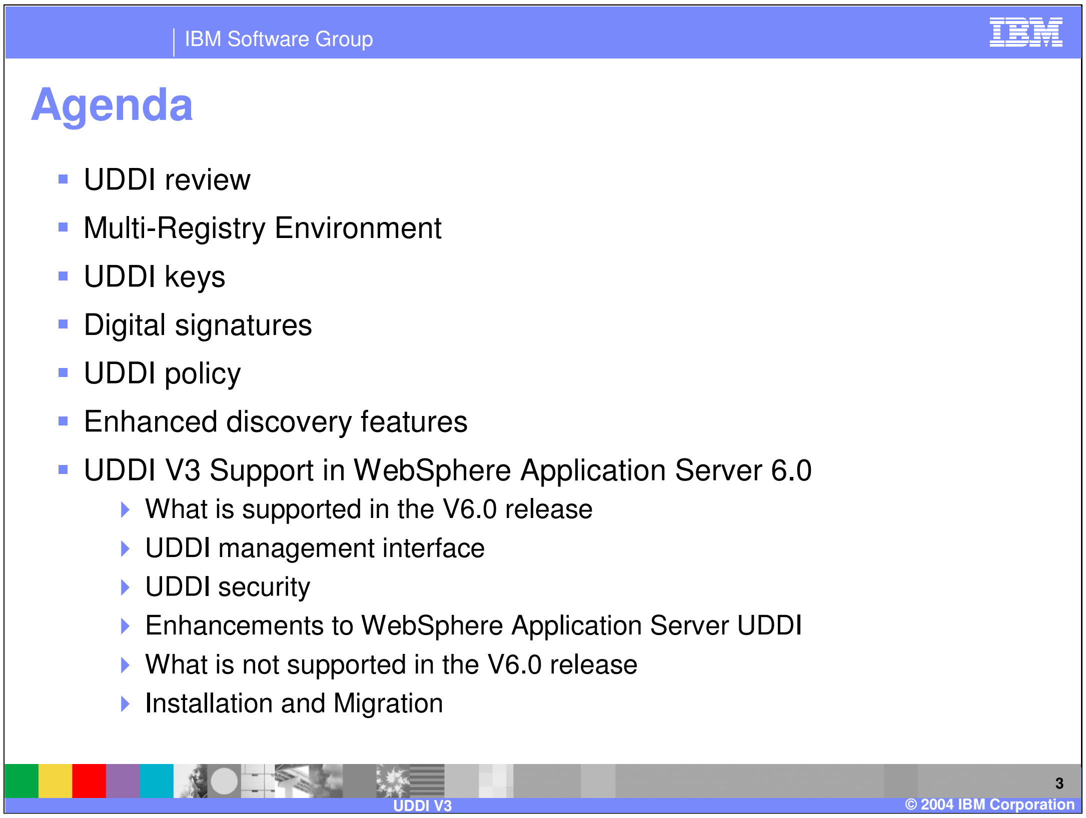
business on demand.

Updated January 25, 2005



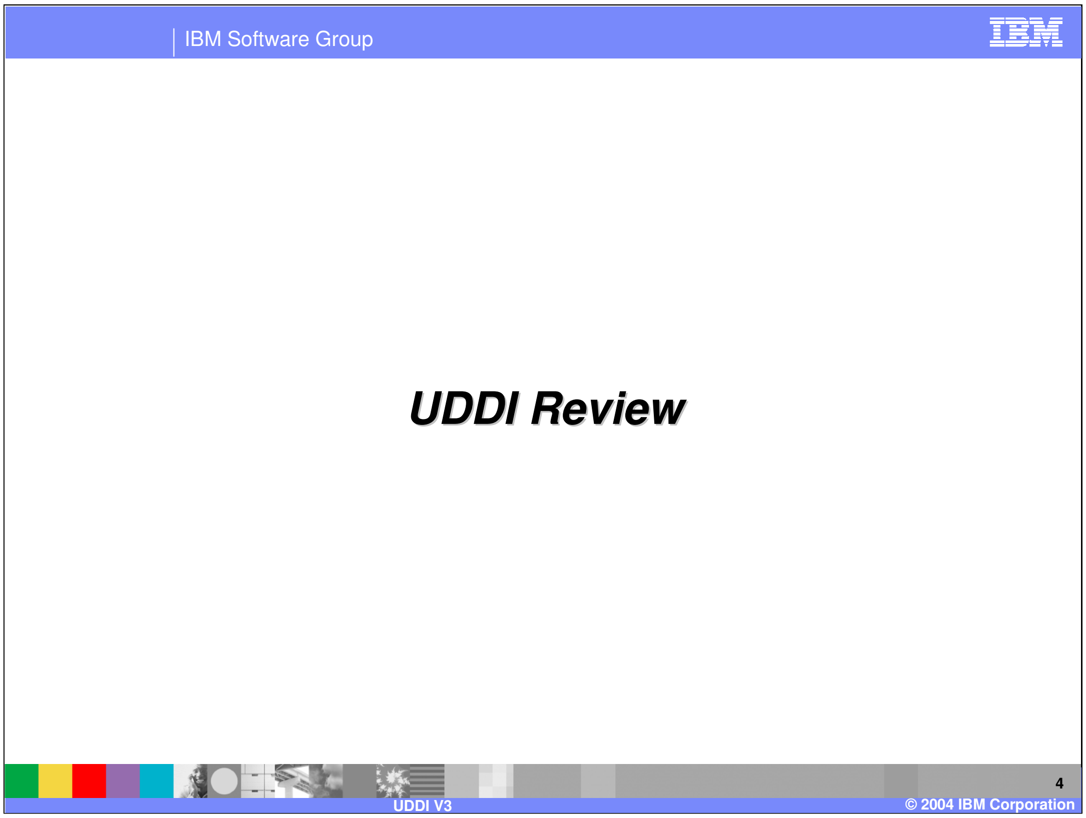
Goals

- Discuss some of the important enhancements in the UDDI V3 specification
- Provide details of how UDDI V3 is implemented in WebSphere Application Server V6
- Describe the UDDI V3 Registry installation process in WebSphere Application Server V6
- Describe the UDDI V2 to UDDI V3 Registry migration process in WebSphere Application Server V6

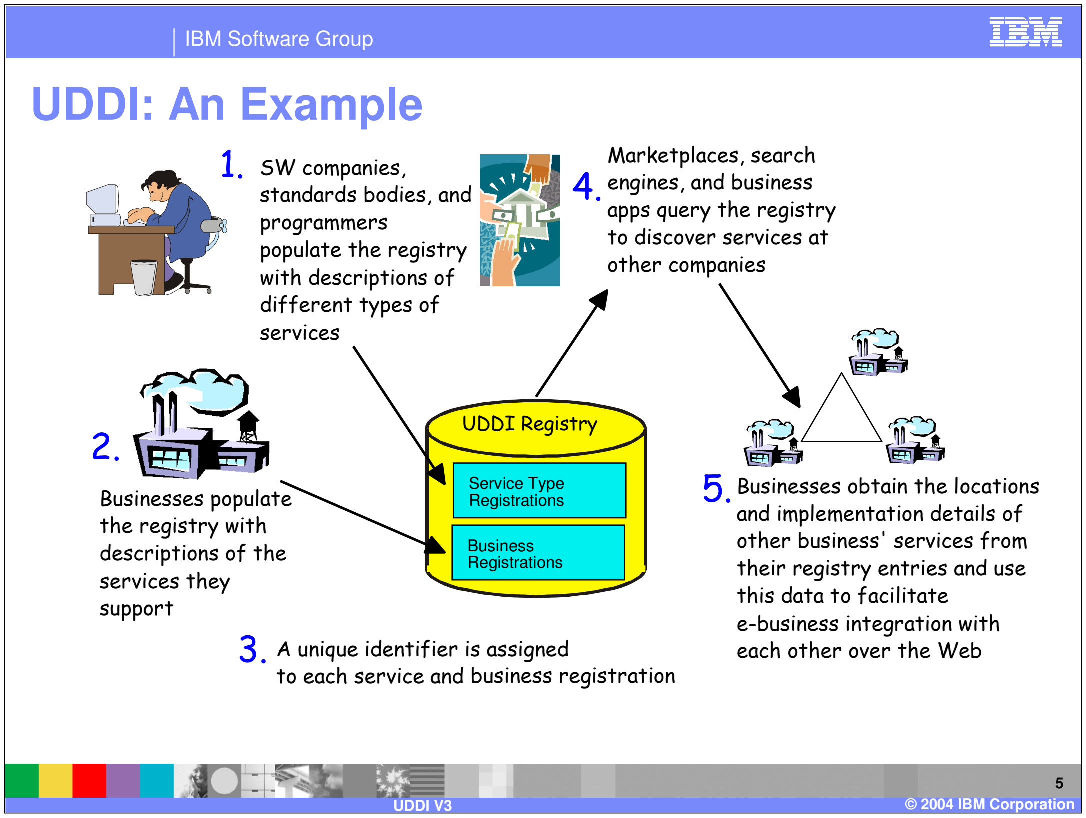


Agenda

- UDDI review
- Multi-Registry Environment
- UDDI keys
- Digital signatures
- UDDI policy
- Enhanced discovery features
- UDDI V3 Support in WebSphere Application Server 6.0
  - What is supported in the V6.0 release
  - UDDI management interface
  - UDDI security
  - Enhancements to WebSphere Application Server UDDI
  - What is not supported in the V6.0 release
  - Installation and Migration



UDDI Review



IBM Software Group

UDDI: An Example

1. SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

2. Businesses populate the registry with descriptions of the services they support

3. A unique identifier is assigned to each service and business registration

4. Marketplaces, search engines, and business apps query the registry to discover services at other companies

5. Businesses obtain the locations and implementation details of other business' services from their registry entries and use this data to facilitate e-business integration with each other over the Web

5

UDDI V3

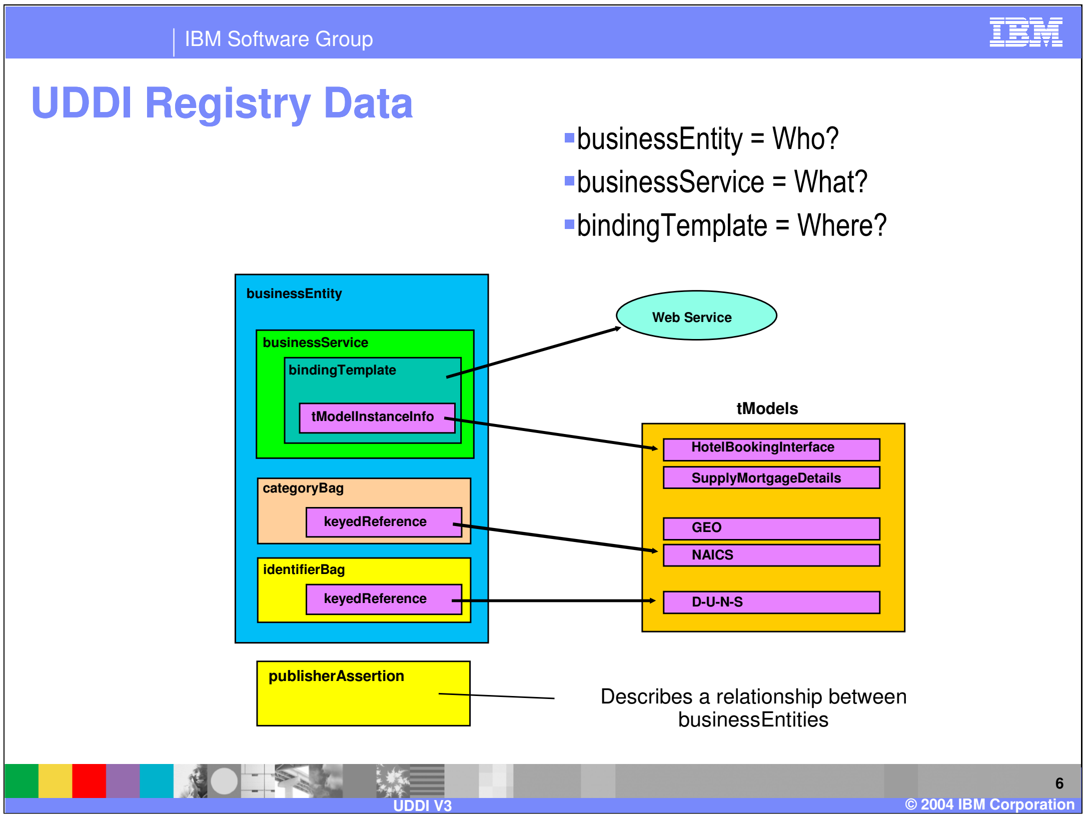
© 2004 IBM Corporation

Businesses populate the registry with descriptions of the services that they support. The registry assigns a unique identifier to each service description and business registration, storing the identifiers in the registry. Service requesters can query the registry to discover services.

In V3, the assignment of the unique identifier may be chosen by the publisher or may be generated programmatically by the Registry.

Important note: whilst this slide represents the vision of UDDI, currently the majority of UDDI registry implementations are **Private**, residing within

intranets, extranets or private networks. These are provided by installing and running a UDDI Registry product, rather than by using the **Universal Business Registry** (UBR), a public registry of nodes operated, currently, by IBM, Microsoft™ and SAP.



UDDI Registry Data

- businessEntity = Who?
- businessService = What?
- bindingTemplate = Where?

businessEntity

businessService

bindingTemplate

tModelInstanceInfo

categoryBag

keyedReference

identifierBag

keyedReference

Web Service

tModels

HotelBookingInterface

SupplyMortgageDetails

GEO

NAICS

D-U-N-S

publisherAssertion

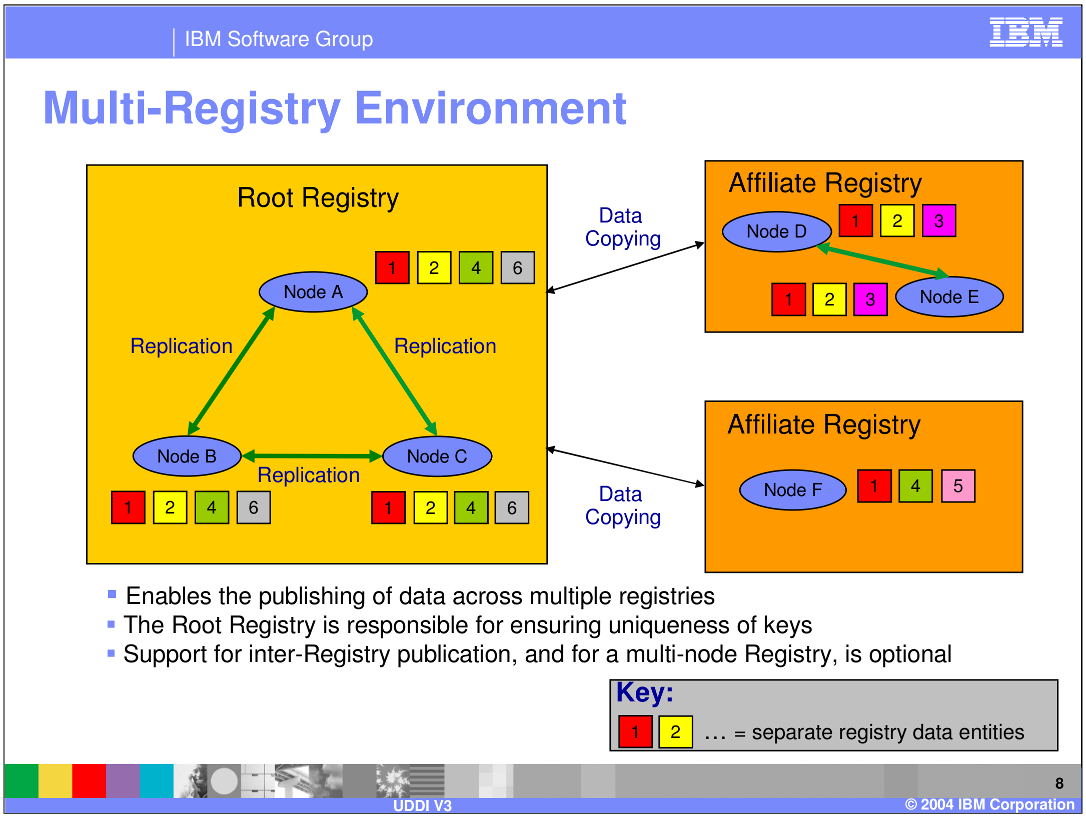
Describes a relationship between businessEntities

tModels are used to represent **standard** interfaces. Many services can register themselves and specify that they conform to a particular tModel. Inquiries can specify that they are only interested in services that conform to a specific tModel.

For example, a company may publish their conformance to a particular EDI standard

tModels are also used as namespace references. A company can qualify their categorization data by reference to the NAICS tModel (which is pre-published in the WebSphere UDDI Registry, along with a number of others)

Multi-Registry Environment



IBM Software Group

Multi-Registry Environment

- Enables the publishing of data across multiple registries
- The Root Registry is responsible for ensuring uniqueness of keys
- Support for inter-Registry publication, and for a multi-node Registry, is optional

Key:
1 2 … = separate registry data entities

8

UDDI V3

A registry is comprised of one or more UDDI nodes. In the example configuration shown, the root registry could be some parent registry, the lower affiliate registry could be a WebSphere V6 registry (the WebSphere UDDI registry is single node only).

The nodes of a registry collectively manage a well-defined set of UDDI data. Typically, this is supported by the use of UDDI replication between the nodes in the registry which reside on different systems (replication was included in the V2 specification).

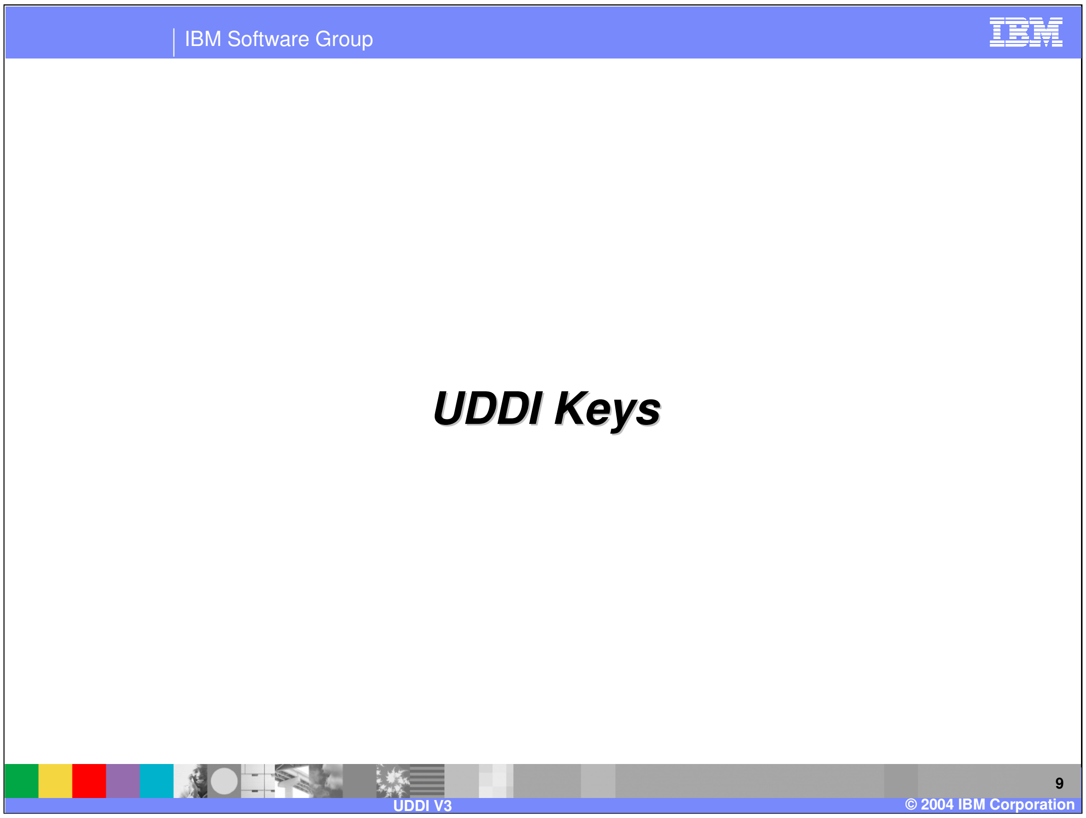
The purpose of a multi-registry environment is to *share* data among the Registries. Specific entities in a Registry, a subset of a Registry, or an entire Registry, may be copied to another Registry.

A root registry serves to delegate key partitions (discussed in more detail in the next section) such that other registries can rely upon the root registry for verification and validation of a given key partition. All other registries that interact with the root registry are called *affiliate* registries. Affiliate registries rely on the root registry to delegate key partitions and insure that uniqueness across key partitions is maintained.

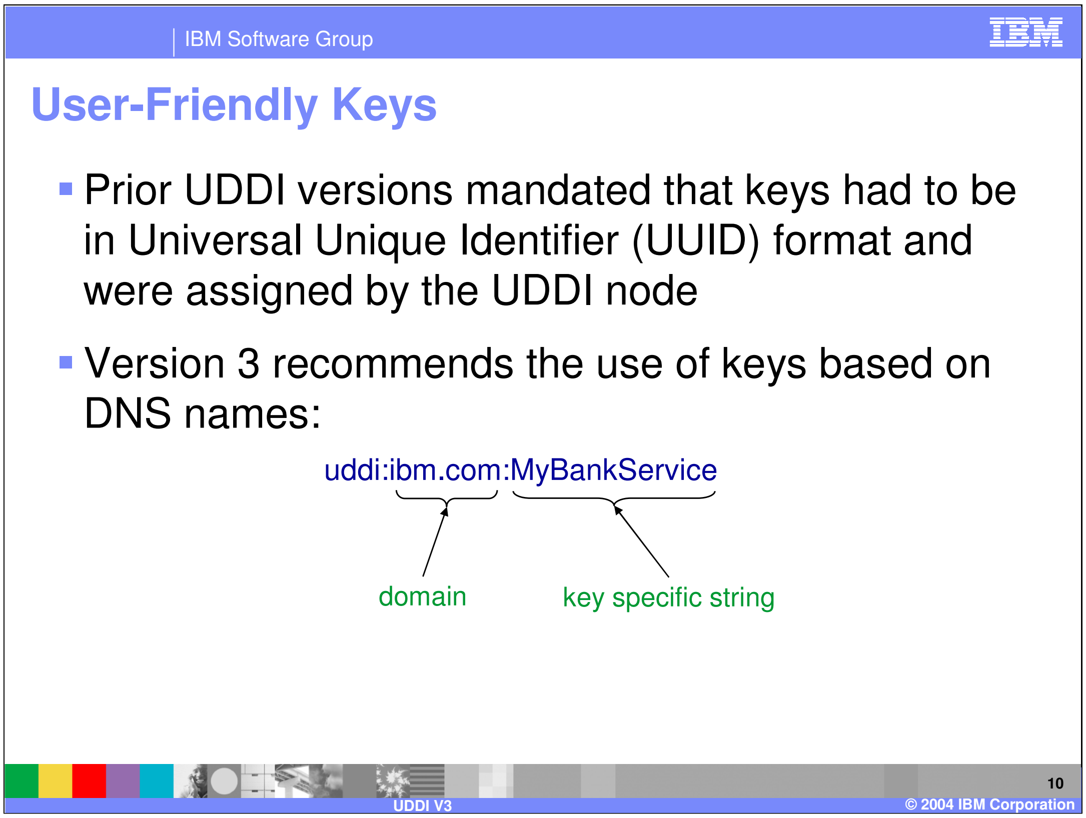
The small coloured rectangles illustrate some of the various possibilities for copying data between the registries. For example:

Date entity 1 has been copied to all three registries.
Data entity 2 is stored on two of the three registries.
Data entity 3 is stored only on one registry.

Within each registry, the data stored on each node is identical



UDDI Keys



User-Friendly Keys

- Prior UDDI versions mandated that keys had to be in Universal Unique Identifier (UUID) format and were assigned by the UDDI node
- Version 3 recommends the use of keys based on DNS names:

uddi:ibm.com:MyBankService

domain

key specific string

UUID keys are still supported.

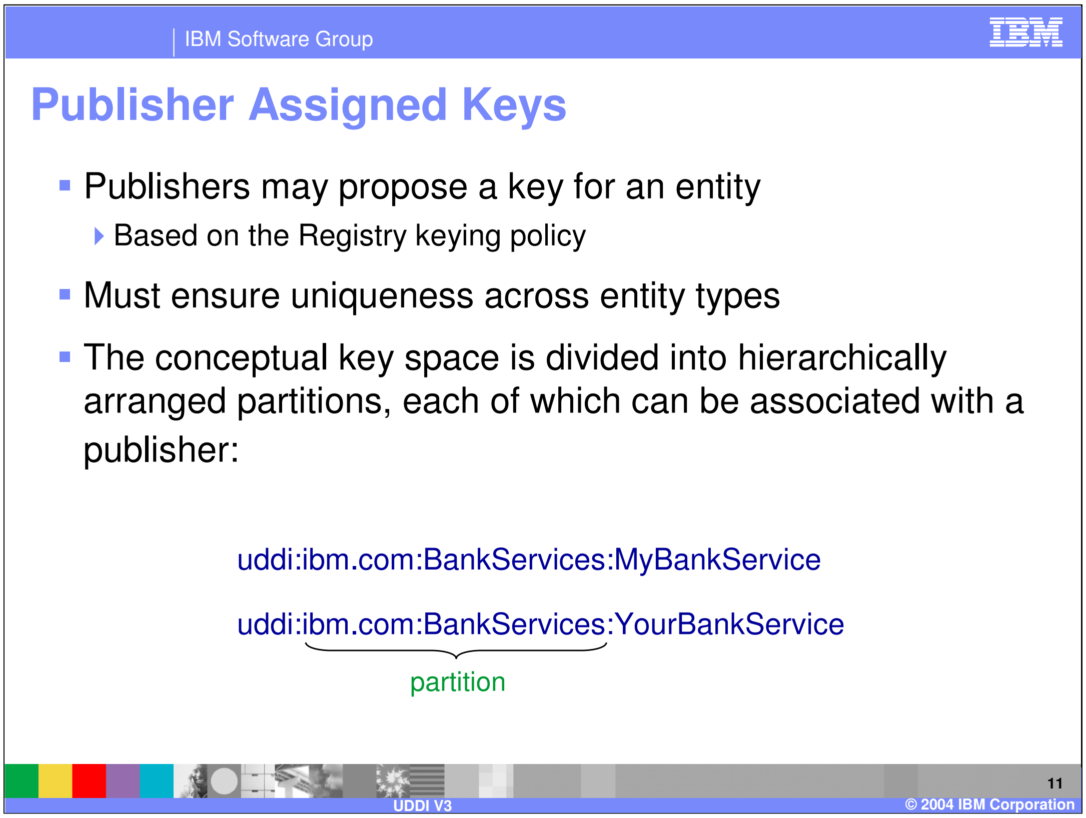
An example of a UUID key is

uddi:4CD7E4BC-648B-426D-9936-443EAAC8AE23

“Sensibly” named keys may be derived from a UUID key:

uddi:4CD7E4BC-648B-426D-9936-443EAAC8AE23:BankService

A single registry may support both key styles side-by-side.



Publisher Assigned Keys

- Publishers may propose a key for an entity
  - Based on the Registry keying policy
- Must ensure uniqueness across entity types
- The conceptual key space is divided into hierarchically arranged partitions, each of which can be associated with a publisher:

uddi:ibm.com:BankServices:MyBankService

uddi:ibm.com:BankServices:YourBankService

partition

If the publisher does not propose a key for an entity, the registry must assign one.

Since entity keys must be unique in a registry without regard to the type of entity and since registries must impose policies concerning which publishers may publish which keys, publisher-assigned keys are subject to rules that UDDI registries enforce.

To ensure that publisher-generated keys do not conflict with one another, registries following the recommended keying scheme assign the authority to generate keys to publishers in the following manner:

1. The conceptual space of uddiKeys is divided into non-overlapping, hierarchically arranged partitions, each of which can be associated with a publisher.

2. Only the publisher associated with a particular partition is given the authority to assign keys within the partition.

3. The publisher with authority for a given partition may designate any publisher it chooses for any partition directly below the partition it manages, provided it has not already designated a publisher to that partition.

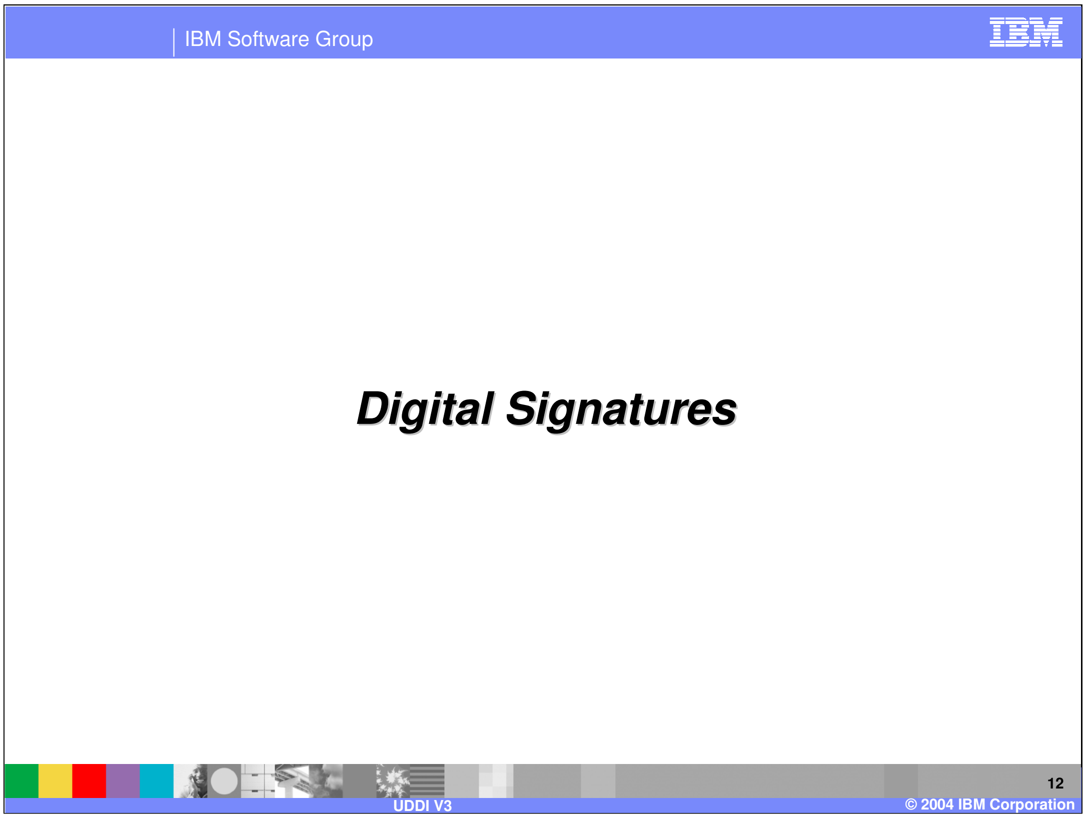
4. The publisher with authority for a partition may transfer its authority to another publisher.

5. Initially, the registry itself has authority for the root partition of the hierarchy.

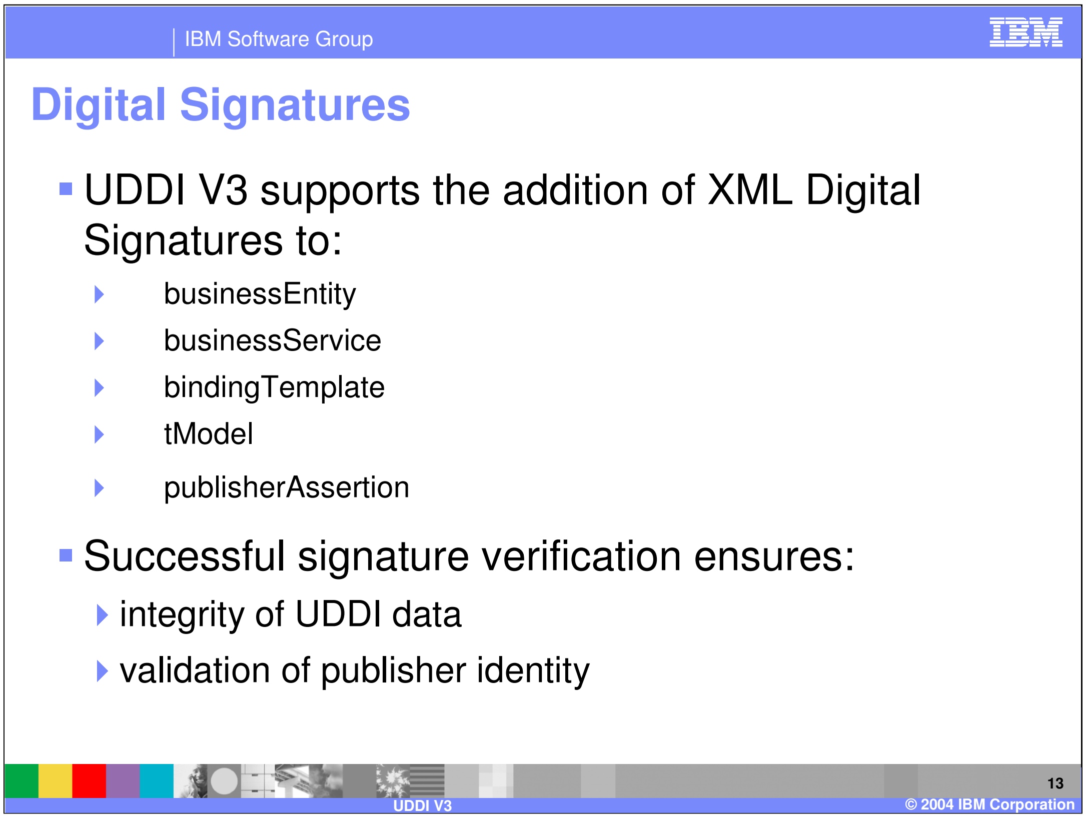
To successfully publish a new entity with a proposed key, the publisher needs to own the key generator tModel for the partition in which the key lies. Typically, a publisher gets ownership by publishing the tModel in question, but publishers can also get ownership in other ways, for example by having another publisher transfer ownership.

Once a publisher owns a key generator tModel that publisher may publish new entities and assign them keys within the key generator tModel’s partition. Publishers are responsible for managing the uniqueness of the keys in the partition they own. If a publisher fails to do so, and generates an already used key, a publish operation could inadvertently replace an entity previously published by that publisher.

A similar mechanism is applied to ensure uniqueness across multiple registries. To bring up each node in a registry affiliated with a root registry, the node should begin by



Digital Signatures



Digital Signatures

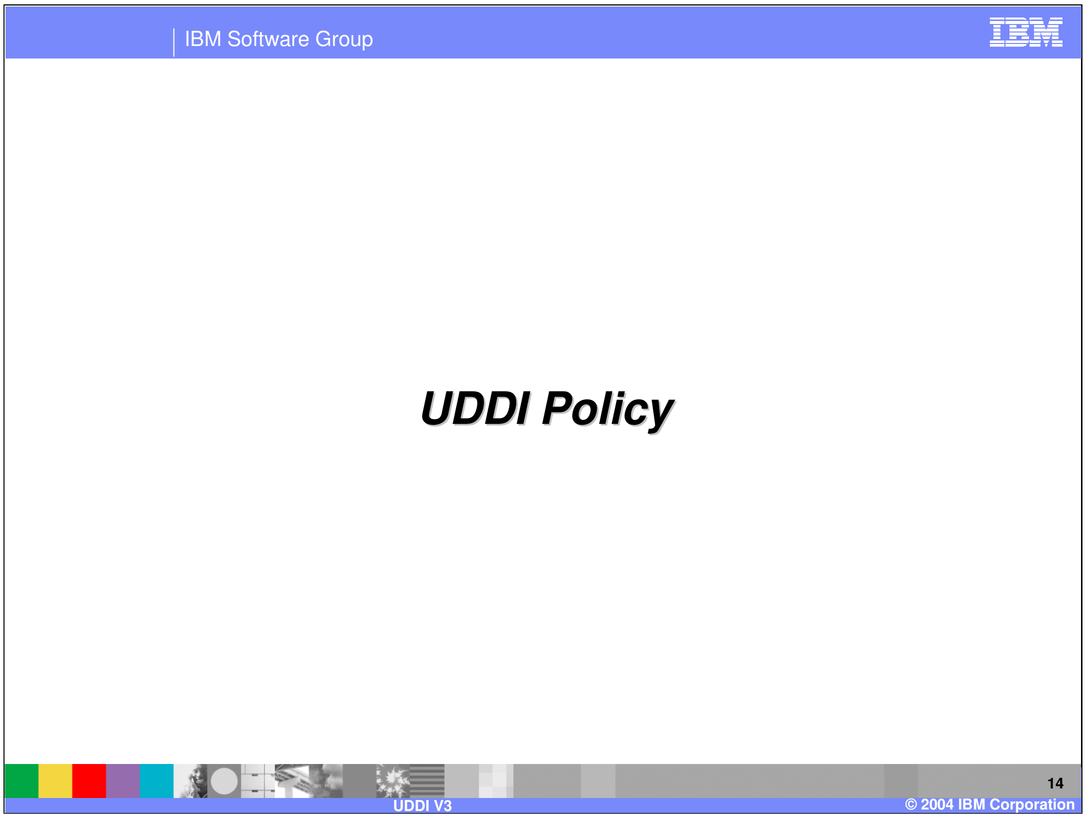
- UDDI V3 supports the addition of XML Digital Signatures to:
  - businessEntity
  - businessService
  - bindingTemplate
  - tModel
  - publisherAssertion
- Successful signature verification ensures:
  - integrity of UDDI data
  - validation of publisher identity

Summary of XML Digital Signature process:

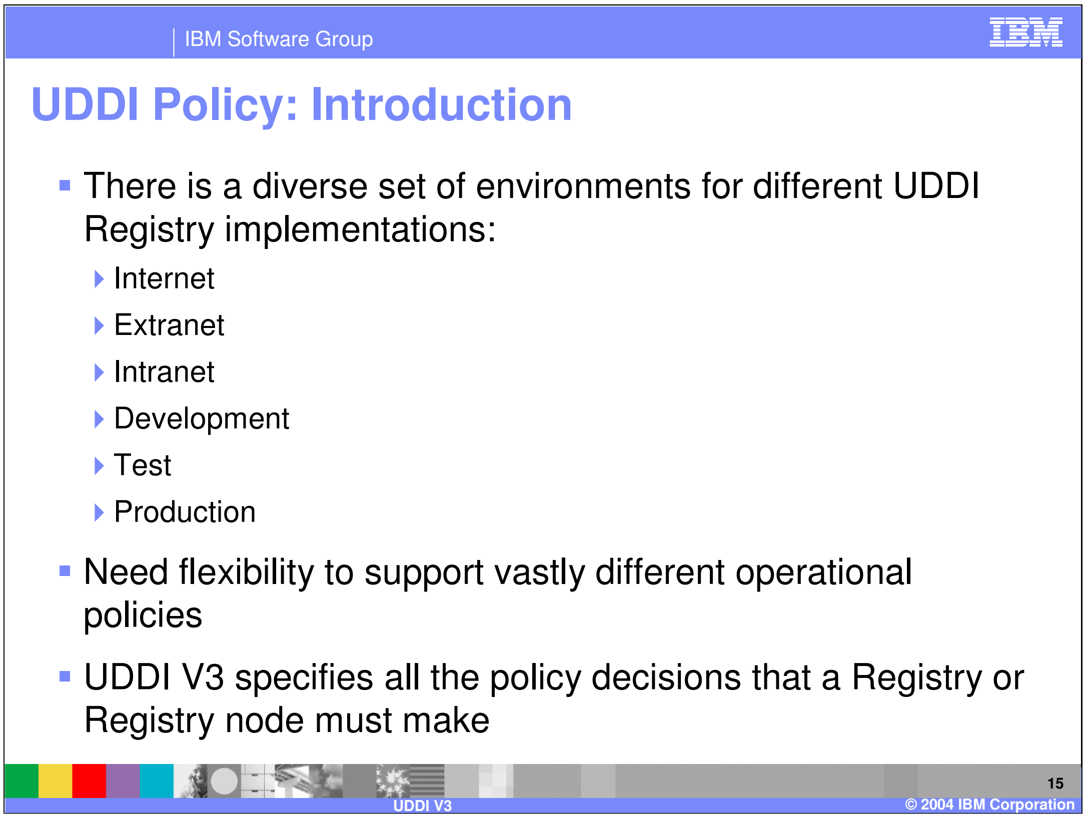
1. Convert the XML to be signed into **Canonical** form (two XML files that are equivalent may have textual differences due, for example, to whitespace, empty xml tags, upper vs lower case, and diacritics such as the cedilla). Canonicalization ensures that such modifications made subsequently to the XML will not invalidate the signature)
2. Possibly apply a further Transform to the XML data. An XPath Transform allows sections of the XML to be omitted from the signature so that subsequent modification of that section will not invalidate the signature. For example, subsequent modification of a BusinessService within a BusinessEntity would invalidate the signature of the BusinessEntity unless the BusinessService is excluded by applying a transform
3. Apply a hashing algorithm (such as SHA1) to the transformed data, D, to produce a **digest** value H(D)
4. Encrypt H(D) using the Publisher's **private key –** this produces the **signature**
5. The Consumer of the data recreates the digest to obtain H'(D), then decrypts the signature using the Publisher's **public key** to reproduce the original H(D). If H(D) = H'(D) then data integrity and signer identity has been verified.

Note: if the publisher does not assign a UDDI key then the generation of the key by the registry node will invalidate the signature. In this event, the specification recommends that publishers generate a signature on elements after all keys in the element and its contained elements have been generated by the node. The generation of a signature where node generated keys are included in the signature is, then, only possible on updates of the data where no new keys are to be generated by the node.

Digital signature will be supported in the WebSphere UDDI V3 Registry and will be packaged in the IBM UDDI V3 Client for Java™ (discussed later).

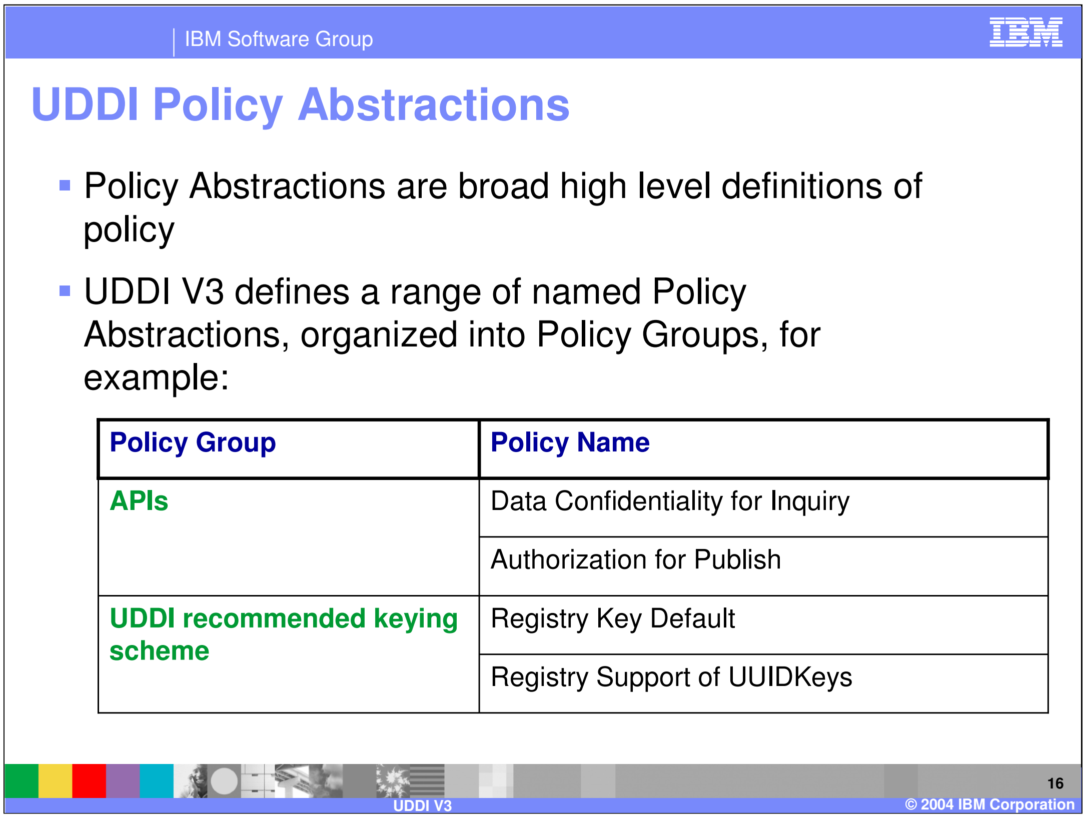


UDDI Policy



UDDI Policy: Introduction

- There is a diverse set of environments for different UDDI Registry implementations:
  - Internet
  - Extranet
  - Intranet
  - Development
  - Test
  - Production
- Need flexibility to support vastly different operational policies
- UDDI V3 specifies all the policy decisions that a Registry or Registry node must make



UDDI Policy Abstractions

- Policy Abstractions are broad high level definitions of policy
- UDDI V3 defines a range of named Policy Abstractions, organized into Policy Groups, for example:

Policy Group	Policy Name
APIs	Data Confidentiality for Inquiry
	Authorization for Publish
UDDI recommended keying scheme	Registry Key Default
	Registry Support of UUIDKeys

APIs

•A registry may specify a policy for the encryption of UDDI data when stored. Furthermore, the data supplied in an API may need to be protected from being “sniffed” on the wire while being transmitted.

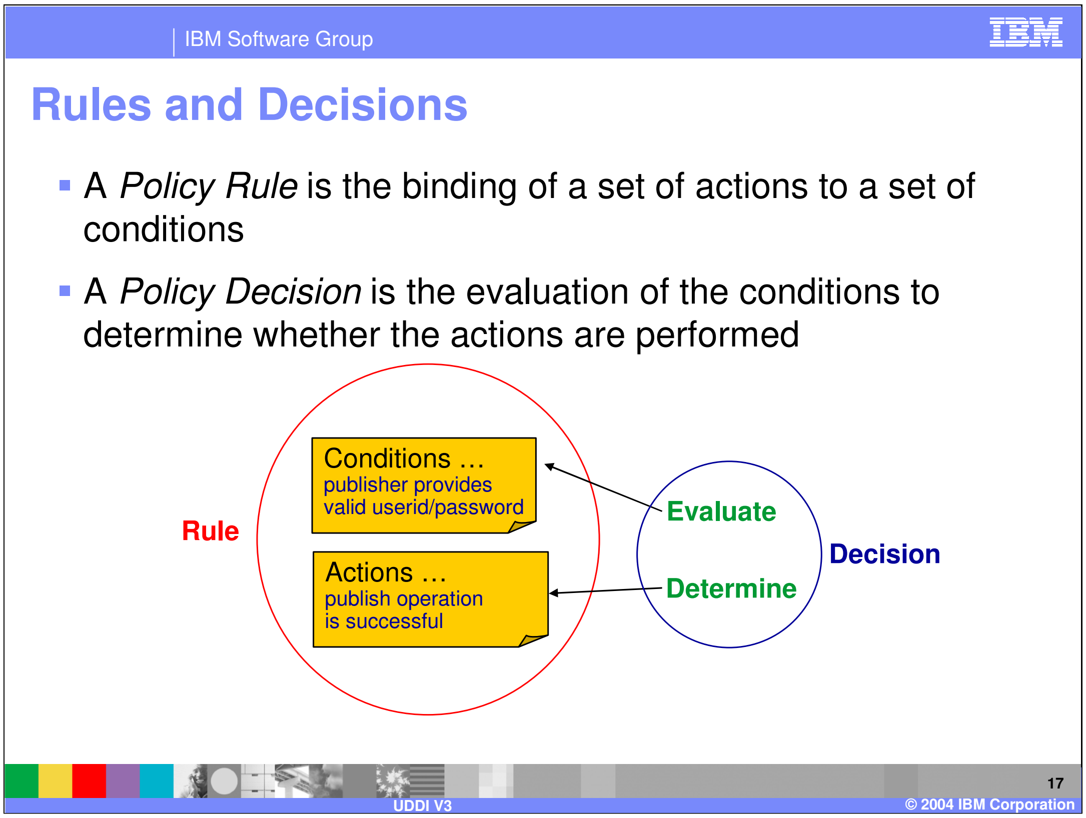
•A registry must have a policy on access to the information registered in it. A registry may specify a policy of global access for all APIs or it may specify a different type of access for each API (for example: Publish, Inquiry).

UDDI recommended keying scheme

•The specification presents a recommended keying scheme that all registries should use.

•The registry must specify what the policy is when a key is not supplied on an API.

•Another policy decision is whether nodes will accept a keyGenerator tModel that is not a domainKey, but is a uuidKey.



Rules and Decisions

- A *Policy Rule* is the binding of a set of actions to a set of conditions
- A *Policy Decision* is the evaluation of the conditions to determine whether the actions are performed

Rule

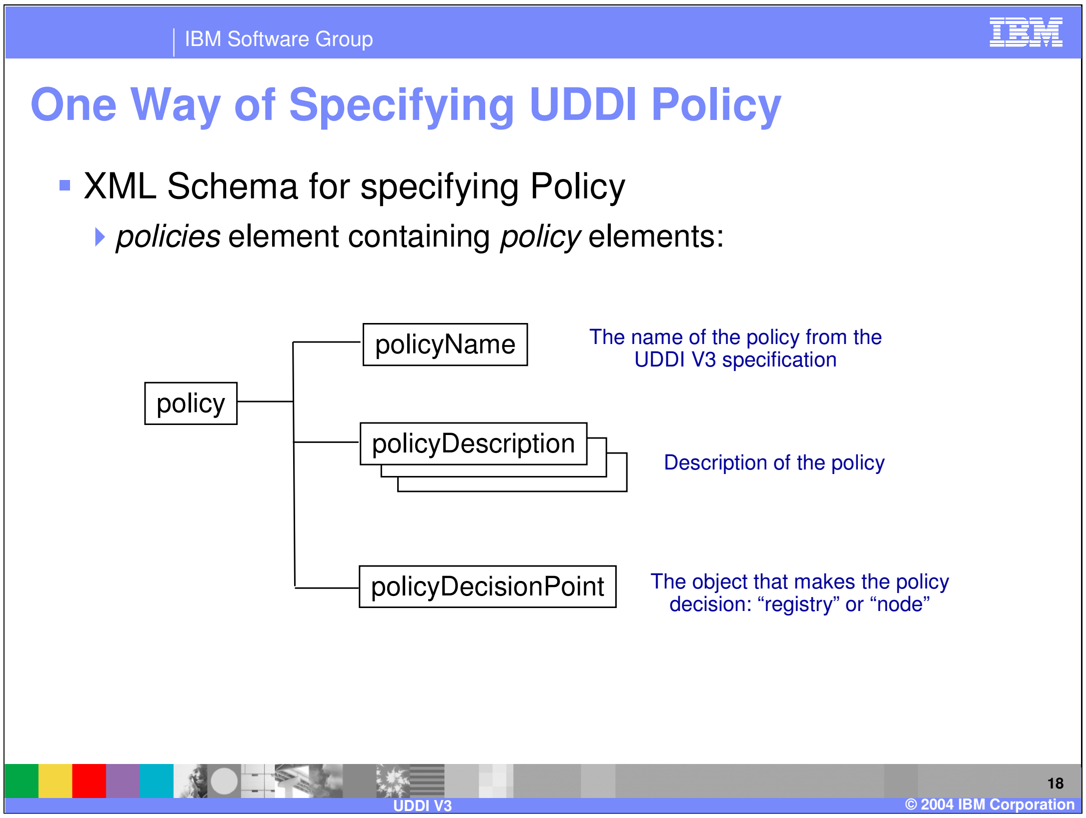
Conditions …
publisher provides
valid userid/password

Actions …
publish operation
is successful

Evaluate

Determine

Decision



IBM Software Group

One Way of Specifying UDDI Policy

- XML Schema for specifying Policy
  - *policies* element containing *policy* elements:

policy

policyName — The name of the policy from the UDDI V3 specification

policyDescription — Description of the policy

policyDecisionPoint — The object that makes the policy decision: “registry” or “node”

18

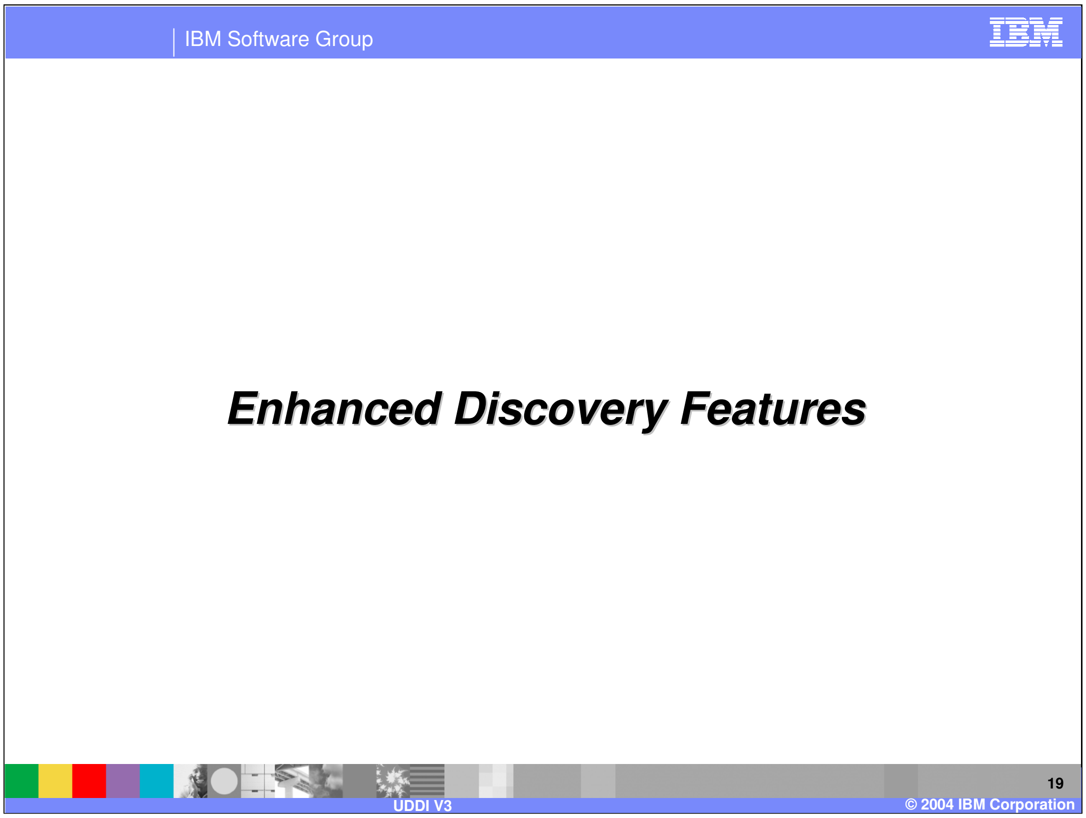
UDDI V3

© 2004 IBM Corporation

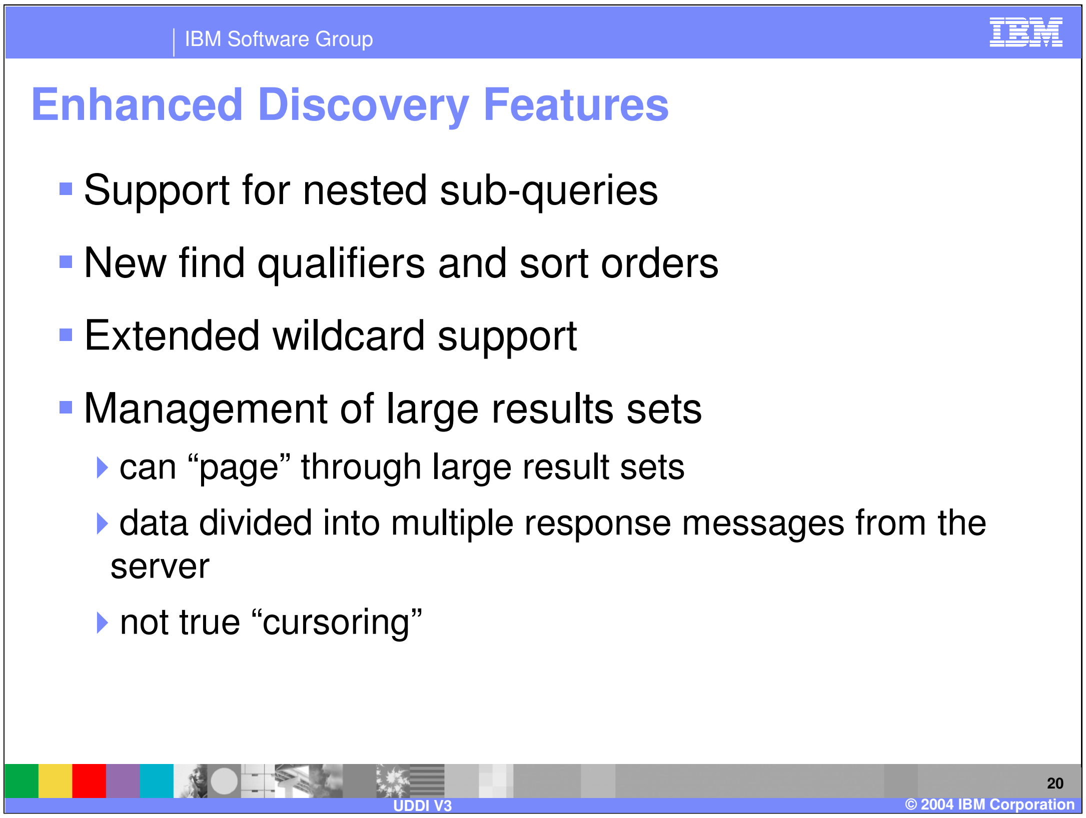
An instance of the policy document should be a Web accessible document and the URL for retrieving this document should be included in the overviewDoc element in the instanceDetails element of a tModelInstanceInfo element referencing a UDDI API set tModel. The contents of the elements in the policy document are intended to be human readable.

The policyDescription element can be adorned with the xml:lang attribute and can appear multiple times to allow for translations of the policy description.

The UDDI Registry in WebSphere V6 specifies policy via the Administrative Console.



Enhanced Discovery Features



IBM Software Group

Enhanced Discovery Features

- Support for nested sub-queries
- New find qualifiers and sort orders
- Extended wildcard support
- Management of large results sets
  - can "page" through large result sets
  - data divided into multiple response messages from the server
  - not true "cursoring"

20

UDDI V3

© 2004 IBM Corporation

Nested sub-queries

By nesting queries for tModels within queries for services, clients can narrow in on the types of services they are searching for much more efficiently

Find qualifiers

For example, "signaturePresent"

approximateMatch – allows for use of wildcard characters like such as "%"

Sort Orders

for example caseInsensitiveSort

Management of large result sets

Several of the inquiry APIs cause a list of results to be returned. In such cases, an element called listDescription may also be returned, containing:

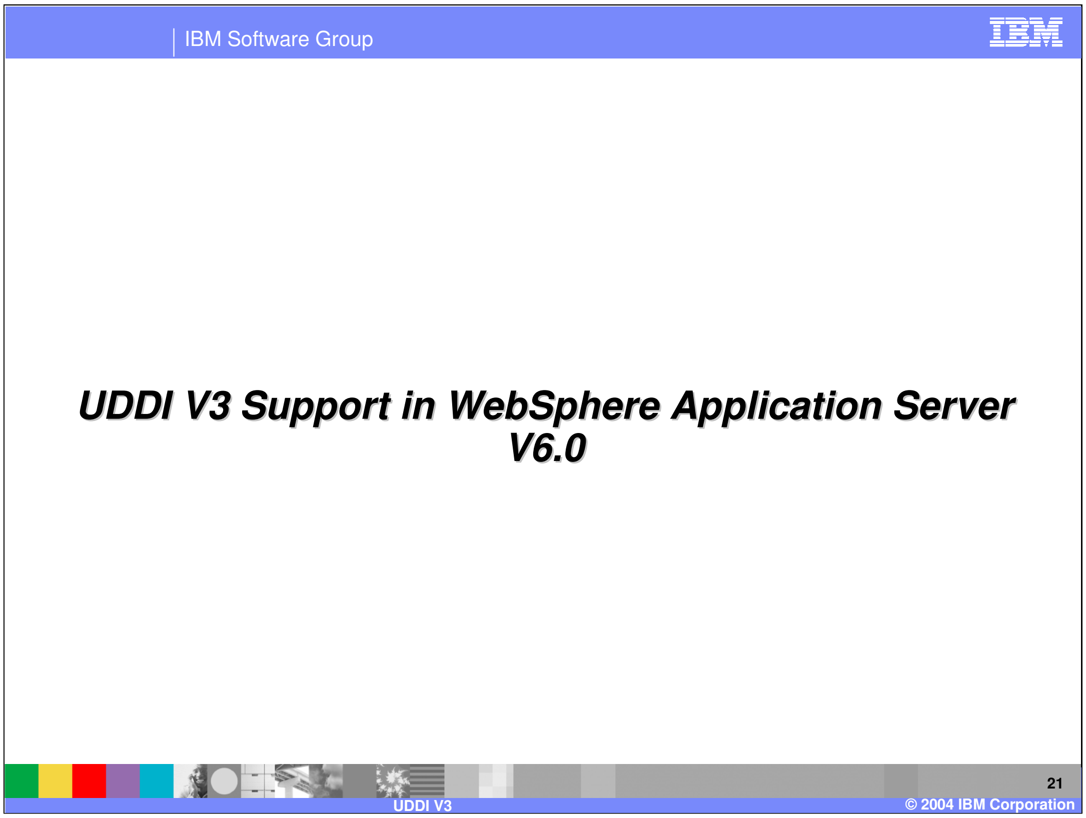
includeCount: is the number of list items returned for the particular response

actualCount: is the number of all available matches at the time this particular query was made

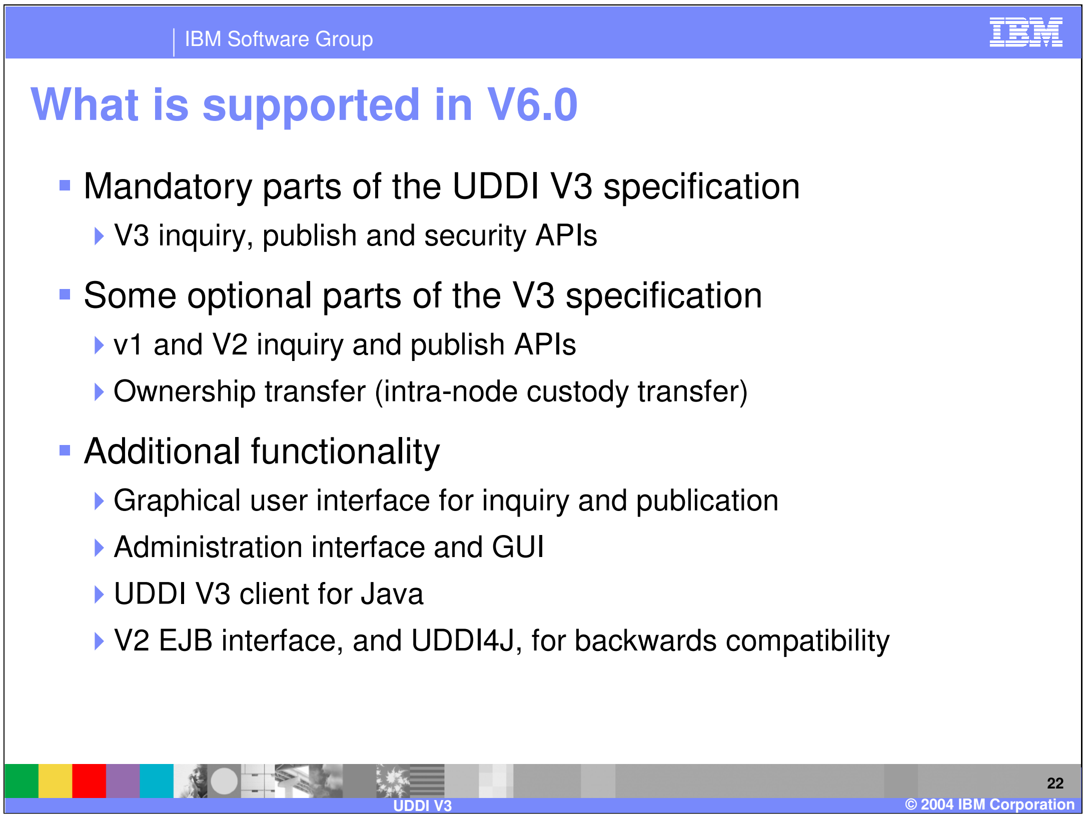
listHead: is an index (with origin of 1) which indicates the index position within all available matches of the first element of the returned result set after any sorting has been applied.

The optional **listHead** argument to a find_xx API may be used to force the list returned to start with a particular element by making further calls. This is useful when the size of the resultant list is too large to be returned in a single query.

"Cursoring" limitations: if the data changes whilst paging through the result sets, then this can result in missed or duplicate data

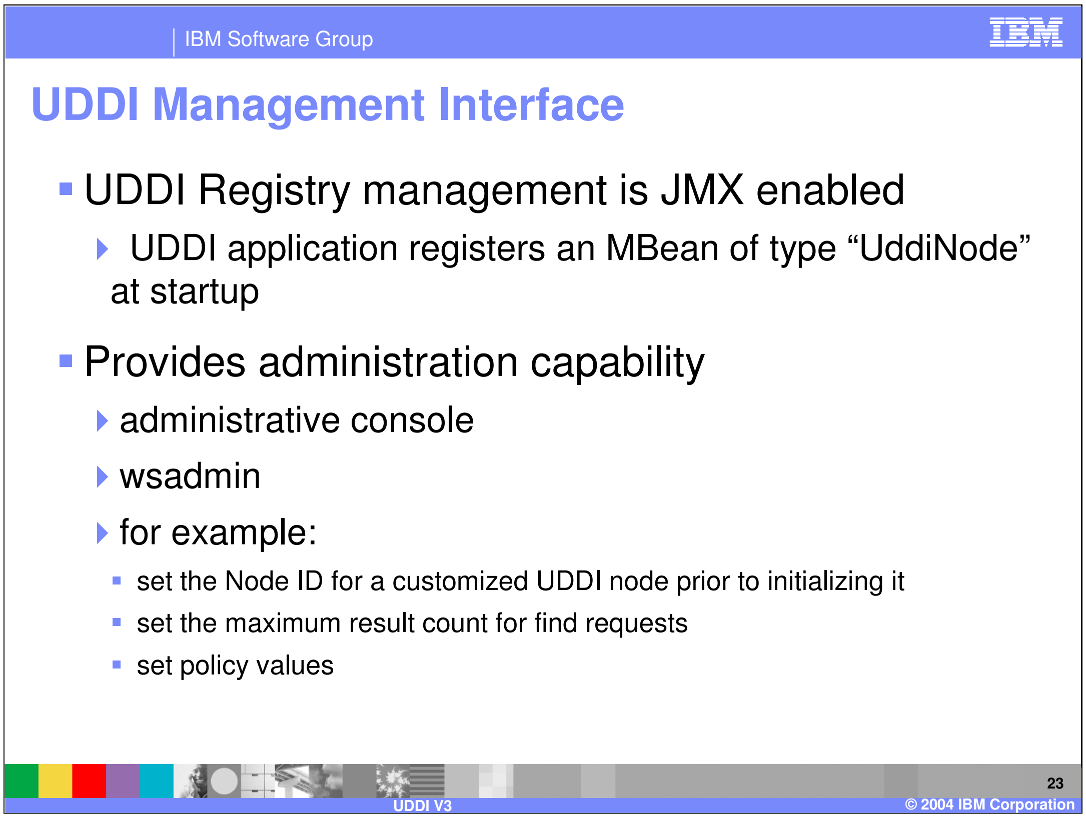


UDDI V3 Support in WebSphere Application Server V6.0



What is supported in V6.0

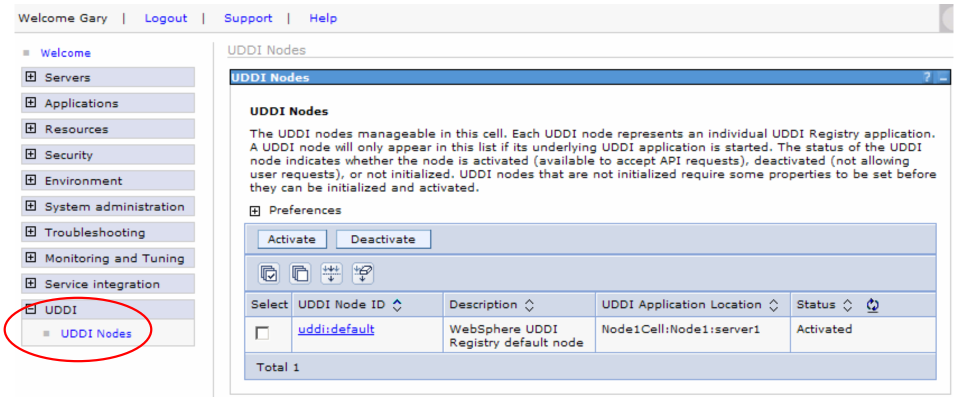
- Mandatory parts of the UDDI V3 specification
  - V3 inquiry, publish and security APIs
- Some optional parts of the V3 specification
  - v1 and V2 inquiry and publish APIs
  - Ownership transfer (intra-node custody transfer)
- Additional functionality
  - Graphical user interface for inquiry and publication
  - Administration interface and GUI
  - UDDI V3 client for Java
  - V2 EJB interface, and UDDI4J, for backwards compatibility



UDDI Management Interface

- UDDI Registry management is JMX enabled
  - UDDI application registers an MBean of type “UddiNode” at startup
- Provides administration capability
  - administrative console
  - wsadmin
  - for example:
    - set the Node ID for a customized UDDI node prior to initializing it
    - set the maximum result count for find requests
    - set policy values

UDDI Nodes Collection



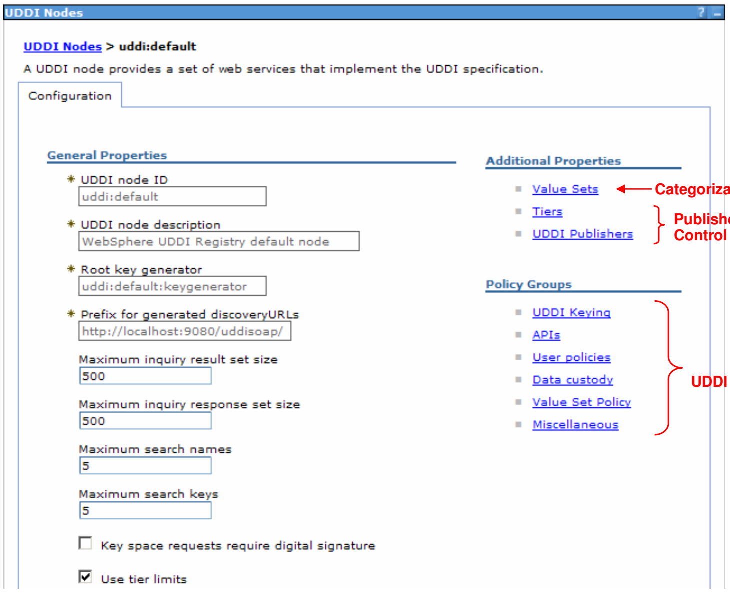
UDDI Nodes

The UDDI nodes manageable in this cell. Each UDDI node represents an individual UDDI Registry application. A UDDI node will only appear in this list if its underlying UDDI application is started. The status of the UDDI node indicates whether the node is activated (available to accept API requests), deactivated (not allowing user requests), or not initialized. UDDI nodes that are not initialized require some properties to be set before they can be initialized and activated.

Select	UDDI Node ID	Description	UDDI Application Location	Status
□	uddi:default	WebSphere UDDI Registry default node	Node1Cell:Node1:server1	Activated

Total 1

General UDDI Node Properties



UDDI Nodes

UDDI Nodes > uddi:default

A UDDI node provides a set of web services that implement the UDDI specification.

Configuration

General Properties

* UDDI node ID: uddi:default

* UDDI node description: WebSphere UDDI Registry default node

* Root key generator: uddi:default:keygenerator

* Prefix for generated discoveryURLs: http://localhost:9080/uddisoap/

Maximum inquiry result set size: 500

Maximum inquiry response set size: 500

Maximum search names: 5

Maximum search keys: 5

□ Key space requests require digital signature

☑ Use tier limits

Additional Properties

- Value Sets ← Categorization Schemes
- Tiers
- UDDI Publishers

} Publisher Access Control

Policy Groups

- UDDI Keying
- APIs
- User policies
- Data custody
- Value Set Policy
- Miscellaneous

} UDDI Policy

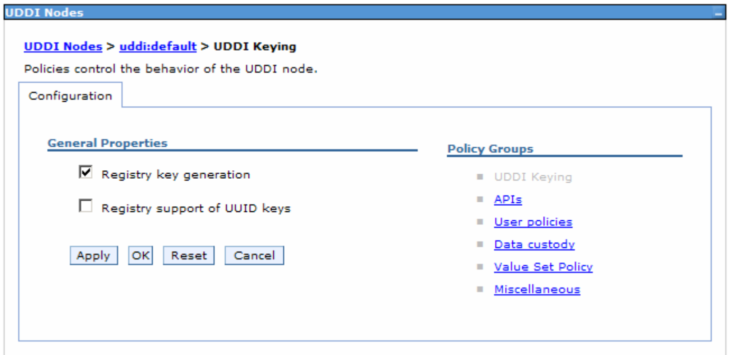
Policy Groups

Refer to settings for different areas of UDDI Policy, as discussed earlier in the presentation

Note that some of the policy choices in the V3 spec are determined by the implementation and so cannot be changed

Publisher Access Control will be discussed shortly

Keying Policy



UDDI Nodes

UDDI Nodes > uddi:default > UDDI Keying

Policies control the behavior of the UDDI node.

Configuration

General Properties

☑ Registry key generation

☐ Registry support of UUID keys

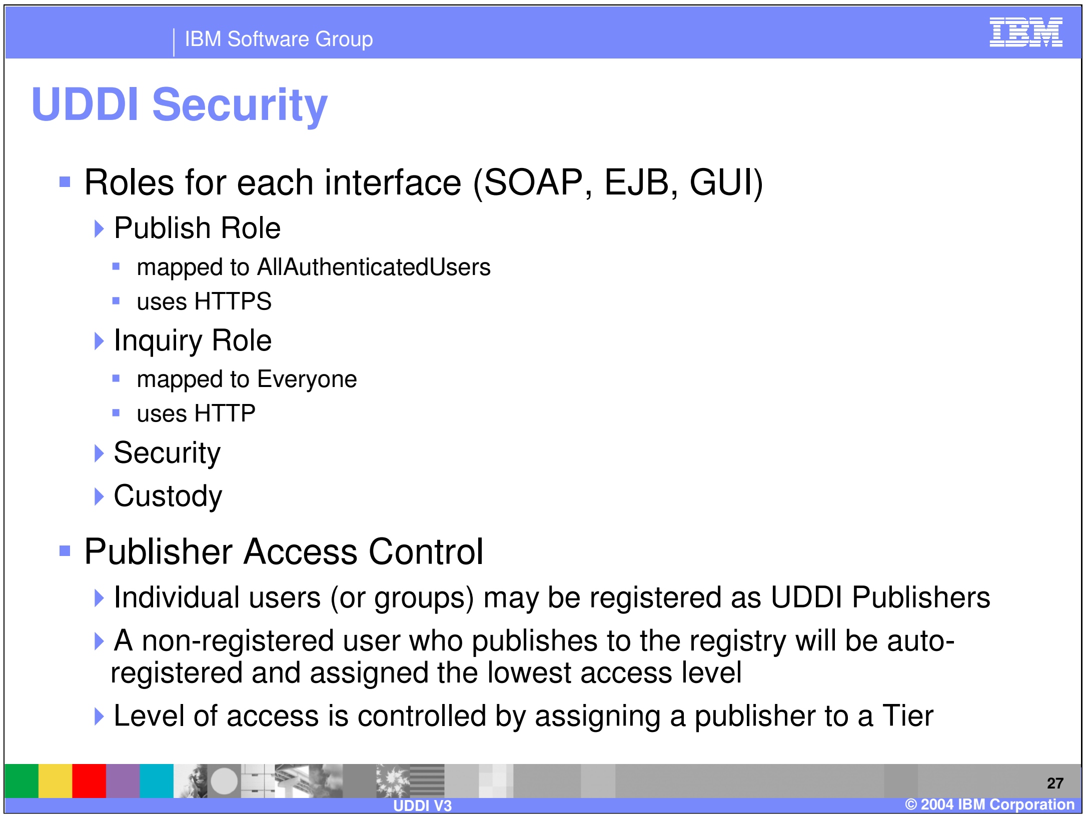
This is one example of UDDI Policy configuration within the Administrative Console

Registry Key Generation

Publishers are allowed to create their own key generator tModels

Registry Support of UUID Keys

Are publishers allowed to specify UUID style keys?



IBM Software Group

UDDI Security

- Roles for each interface (SOAP, EJB, GUI)
  - Publish Role
    - mapped to AllAuthenticatedUsers
    - uses HTTPS
  - Inquiry Role
    - mapped to Everyone
    - uses HTTP
  - Security
  - Custody
- Publisher Access Control
  - Individual users (or groups) may be registered as UDDI Publishers
  - A non-registered user who publishes to the registry will be auto-registered and assigned the lowest access level
  - Level of access is controlled by assigning a publisher to a Tier

UDDI V3

27

© 2004 IBM Corporation

Role security applies if WebSphere security is enabled. Mappings and SSL usage specified above are default settings but can be changed through the Administrative Console.

UDDI Publishers

Administrators may add named publishers and assign them to a Tier, thus controlling the amount of Registry entries which that publisher is allowed to create. In addition, the publisher may be granted/denied the following entitlements:

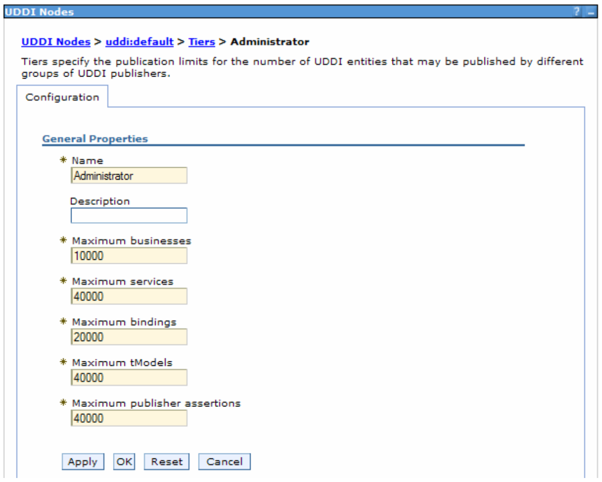
Allowed to publish keyGenerator with derived keys
Allowed to publish keyGenerator with domain keys
Allowed to publish keyGenerator
Allowed to publish with UUID key
Allowed to publish keyGenerator with UUID keys

Tiers are discussed on the next slide

IBM Software Group

IBM

UDDI Security: Tiers



UDDI Nodes

[UDDI Nodes](#) > [uddi:default](#) > [Tiers](#) > Administrator

Tiers specify the publication limits for the number of UDDI entities that may be published by different groups of UDDI publishers.

Configuration

General Properties

* Name
Administrator

Description

* Maximum businesses
10000

* Maximum services
40000

* Maximum bindings
20000

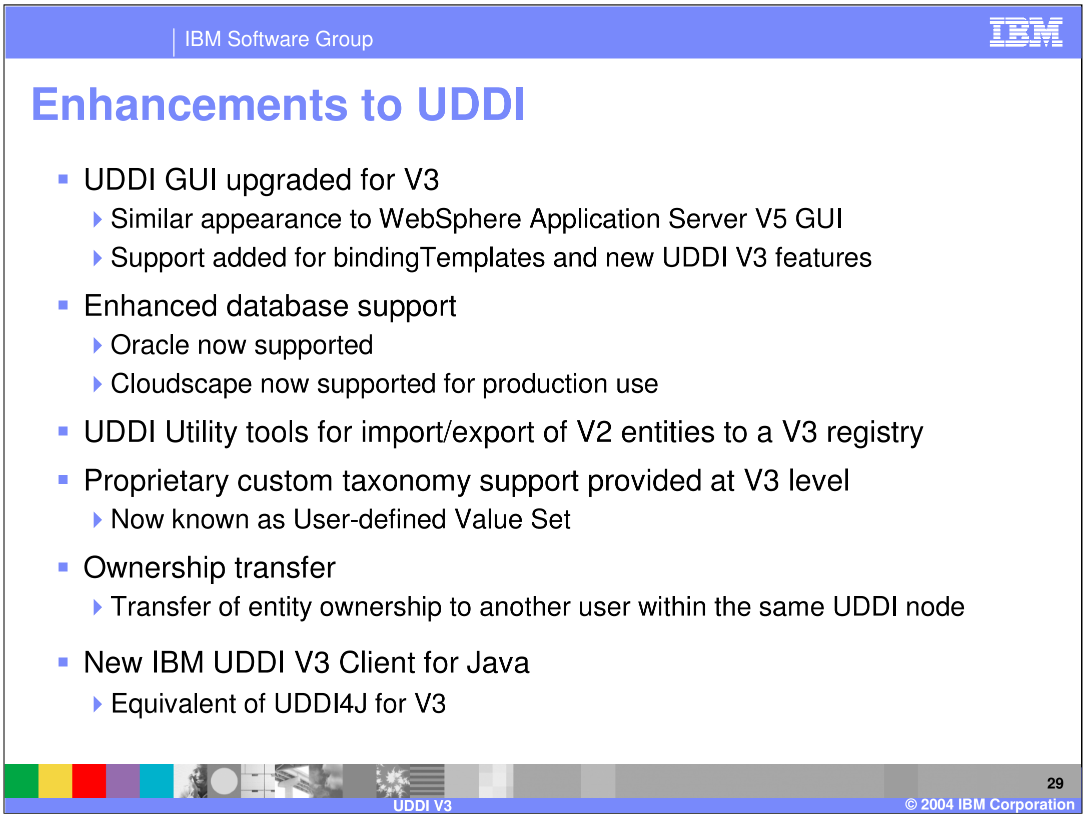
* Maximum tModels
40000

* Maximum publisher assertions
40000

Tiers

Administrators may create named Tiers, and then specify the maximum numbers of Businesses, Services, Bindings, tModels and Publisher Assertions that a publisher assigned to that Tier may create in the registry. Five Tiers are provided by default (Tier0, Tier1, Tier2, Tier3, Tier4)

The above screen capture shows the default settings for the Administrator Tier



Enhancements to UDDI

- UDDI GUI upgraded for V3
  - Similar appearance to WebSphere Application Server V5 GUI
  - Support added for bindingTemplates and new UDDI V3 features
- Enhanced database support
  - Oracle now supported
  - Cloudscape now supported for production use
- UDDI Utility tools for import/export of V2 entities to a V3 registry
- Proprietary custom taxonomy support provided at V3 level
  - Now known as User-defined Value Set
- Ownership transfer
  - Transfer of entity ownership to another user within the same UDDI node
- New IBM UDDI V3 Client for Java
  - Equivalent of UDDI4J for V3

▪Level of support provided by GUI is similar to that provided in V2. As before, the UDDI GUI is intended for familiarisation with UDDI structures, and for finding data; complex publications are best achieved programmatically.

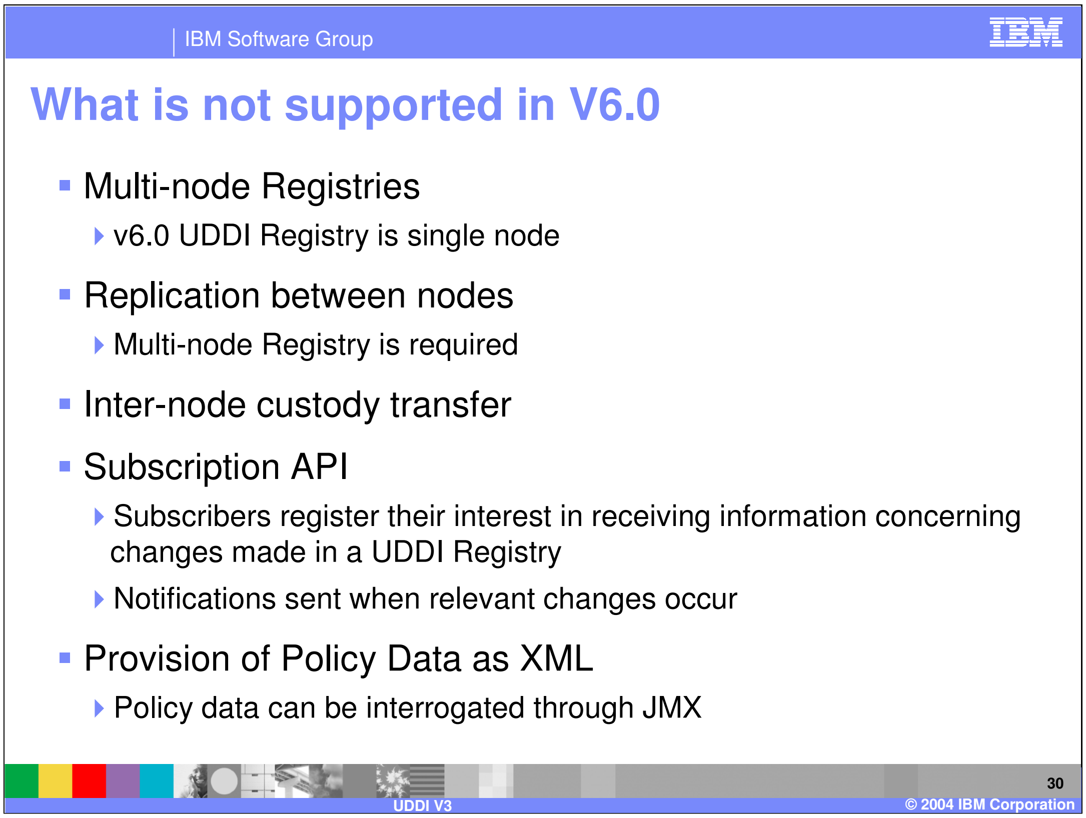
▪The set of supported databases will be further extended in a later V6 release

▪IBM UDDI V3 Client for Java is a JAX-RPC style client based on Axis, and is the V3 equivalent of UDDI4J. UDDI4J is still supported, but deprecated.

▪Utility tools were introduced in V5.1 to allow data to be exported from one registry and imported into another. This is for import/export of V2 entities from a V2 or V3 registry to a V3 registry. A tool for import/export of V3 entities is not needed as the V3 registry supports adding entities with a supplied key via the normal V3 API

▪Custom taxonomy support, allowing users to create their own categorization schemes or value sets, was introduced in V5.0.2. “Value Set” is the terminology used in the V3 specification

Note that the WebSphere V3 registry will still accept V2 requests (and v1)



IBM Software Group

What is not supported in V6.0

- Multi-node Registries
  - v6.0 UDDI Registry is single node
- Replication between nodes
  - Multi-node Registry is required
- Inter-node custody transfer
- Subscription API
  - Subscribers register their interest in receiving information concerning changes made in a UDDI Registry
  - Notifications sent when relevant changes occur
- Provision of Policy Data as XML
  - Policy data can be interrogated through JMX

30

UDDI V3

© 2004 IBM Corporation

There does not currently appear to be much demand for **Private** multi-node registries

The subscription API allows monitoring of activity in a registry by registering to track new, changed and deleted entries for each of these entities:

•businessEntity

•businessService

•bindingTemplate

•tModel

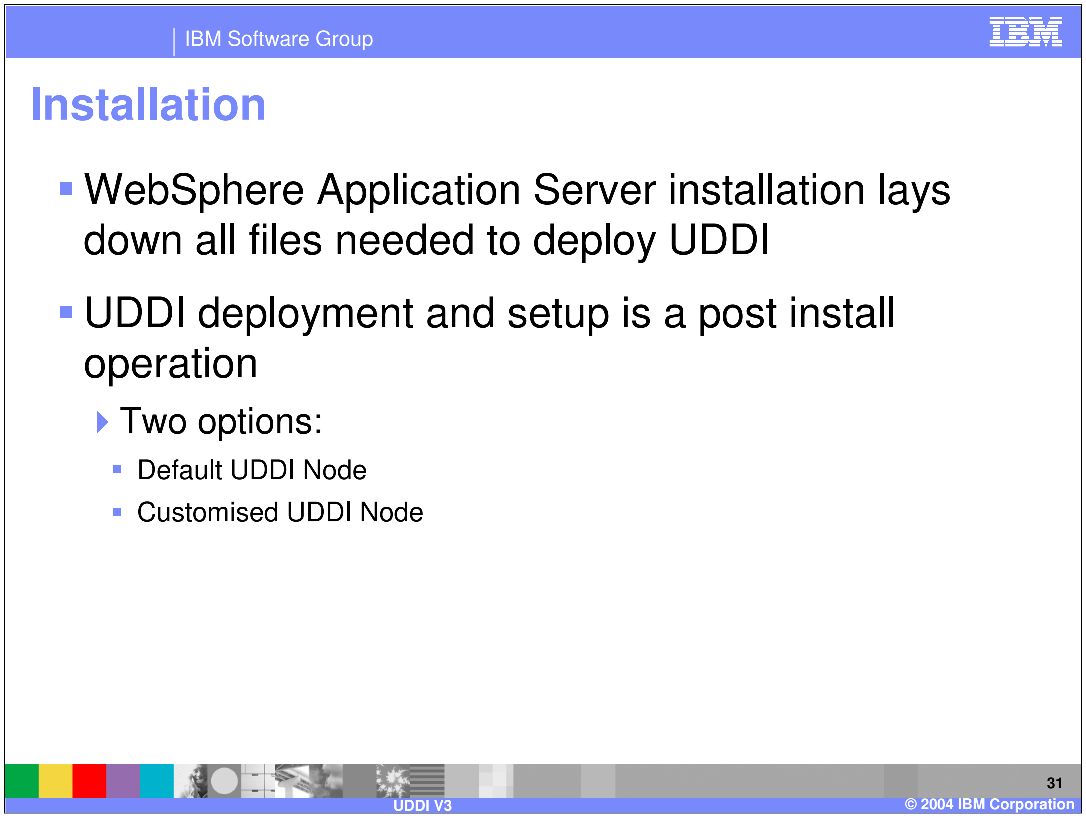
•related businessEntity

•publisherAssertion (limited to those publisherAssertions for which the subscriber owns at least one of the businesses referenced)

Two notification patterns are defined. Nodes may support either or both:

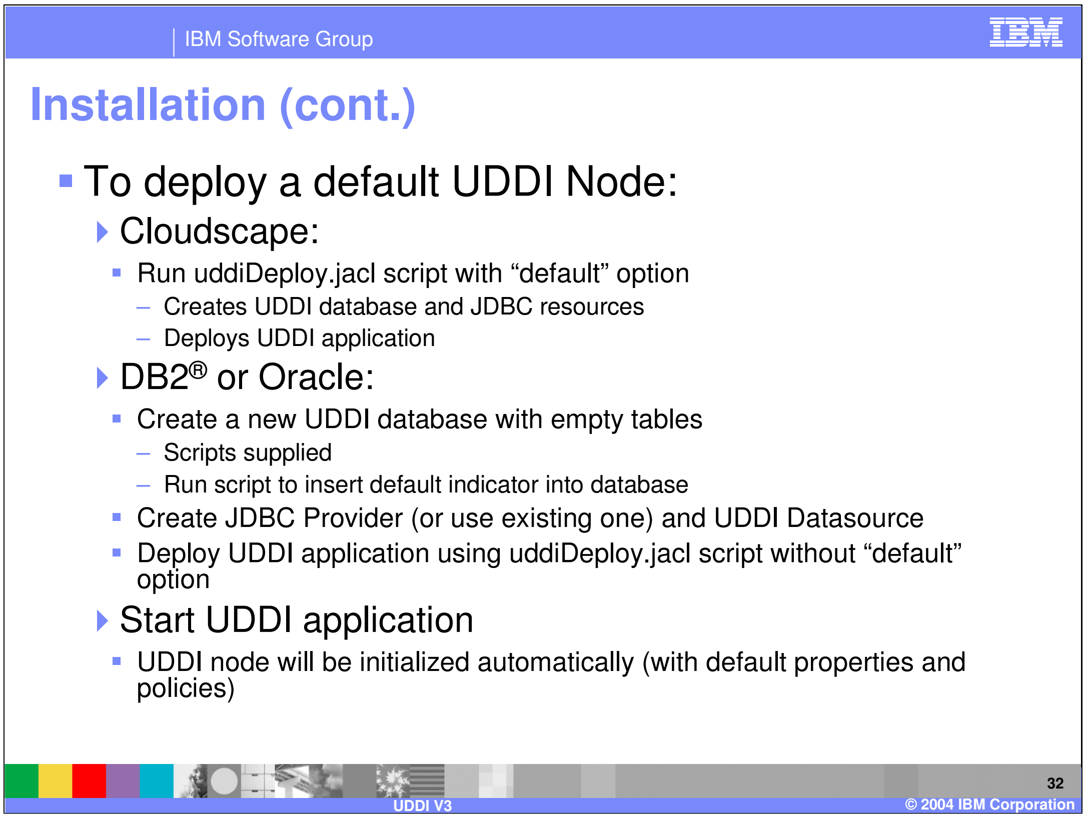
•Asynchronous notification – subscribers choose to be asynchronously notified by the node when registry data of interest changes via calls to the notify_subscriptionListener API, which they implement as a “subscription listener” service.

•Synchronous change tracking – subscribers issue a synchronous request using the get_subscriptionResults API to obtain information on activity in the registry which matches their subscription preferences.



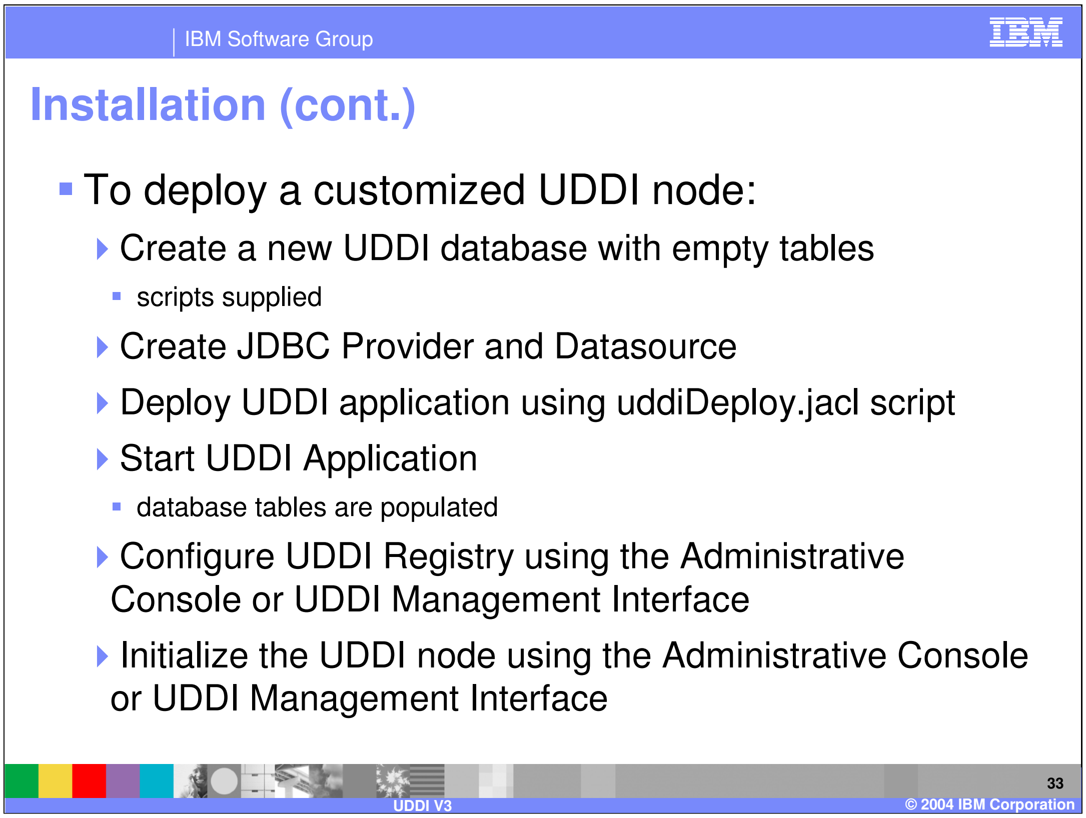
Installation

- WebSphere Application Server installation lays down all files needed to deploy UDDI
- UDDI deployment and setup is a post install operation
  - Two options:
    - Default UDDI Node
    - Customised UDDI Node



Installation (cont.)

- To deploy a default UDDI Node:
  - Cloudscape:
    - Run uddiDeploy.jacl script with “default” option
      - Creates UDDI database and JDBC resources
      - Deploys UDDI application
  - DB2® or Oracle:
    - Create a new UDDI database with empty tables
      - Scripts supplied
      - Run script to insert default indicator into database
    - Create JDBC Provider (or use existing one) and UDDI Datasource
    - Deploy UDDI application using uddiDeploy.jacl script without “default” option
  - Start UDDI application
    - UDDI node will be initialized automatically (with default properties and policies)



IBM Software Group

Installation (cont.)

- To deploy a customized UDDI node:
  - Create a new UDDI database with empty tables
    - scripts supplied
  - Create JDBC Provider and Datasource
  - Deploy UDDI application using uddiDeploy.jacl script
  - Start UDDI Application
    - database tables are populated
  - Configure UDDI Registry using the Administrative Console or UDDI Management Interface
  - Initialize the UDDI node using the Administrative Console or UDDI Management Interface

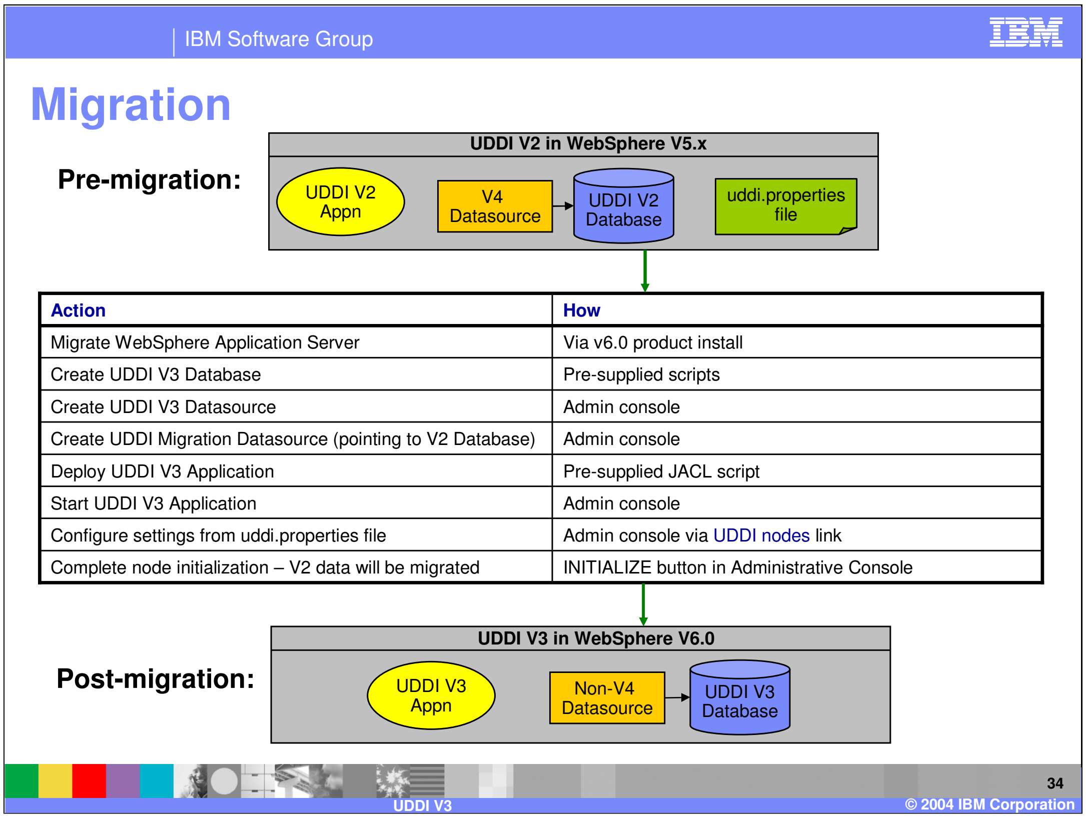
UDDI V3

33

Instructions explaining how to create the database and call the SQL scripts are provided in the Information Center.

A uddiRemove.jacl script is provided which will remove the application. uddiDeploy will remove an existing application in any case and then redeploy.

However, if you already have a registry and want to start again, it is not necessary to redeploy the application. The application looks, at startup, to see if the database is empty and, if it is, repopulates it with initialization data



IBM Software Group

Migration

Pre-migration:

UDDI V2 in WebSphere V5.x

UDDI V2 Appn

V4 Datasource

UDDI V2 Database

uddi.properties file

Action	How
Migrate WebSphere Application Server	Via v6.0 product install
Create UDDI V3 Database	Pre-supplied scripts
Create UDDI V3 Datasource	Admin console
Create UDDI Migration Datasource (pointing to V2 Database)	Admin console
Deploy UDDI V3 Application	Pre-supplied JACL script
Start UDDI V3 Application	Admin console
Configure settings from uddi.properties file	Admin console via UDDI nodes link
Complete node initialization – V2 data will be migrated	INITIALIZE button in Administrative Console

Post-migration:

UDDI V3 in WebSphere V6.0

UDDI V3 Appn

Non-V4 Datasource

UDDI V3 Database

34

UDDI V3

© 2004 IBM Corporation

A fully automated migration process is not provided, as there are some configuration actions and decisions that need to be taken in order to ensure correct migration. There are significant differences between V2 and V3 which need to be considered carefully before migration.

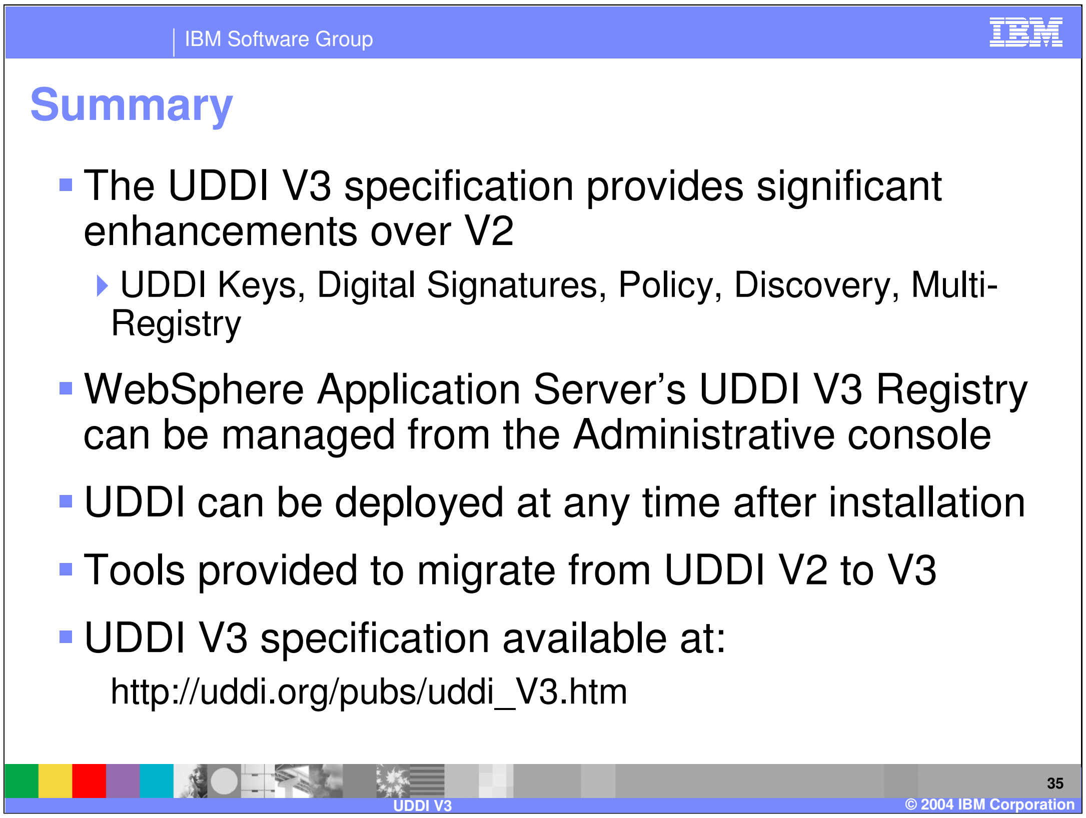
There is no longer a uddi.properties file in V3 – UDDI properties are controlled by UDDI management

The UUDI V2 datasource is a V4 datasource. The UDDI V3 datasource, and the migration datasource, are both non-v4.

The UDDI Node Initialization process will detect that there is a UDDI migration datasource, and will therefore migrate the UDDI V2 data as part of the UDDI node initialization processing.

In order for the initialization process to locate the migration datasource, it must be given a JNDI name of datasources/uddimigration.

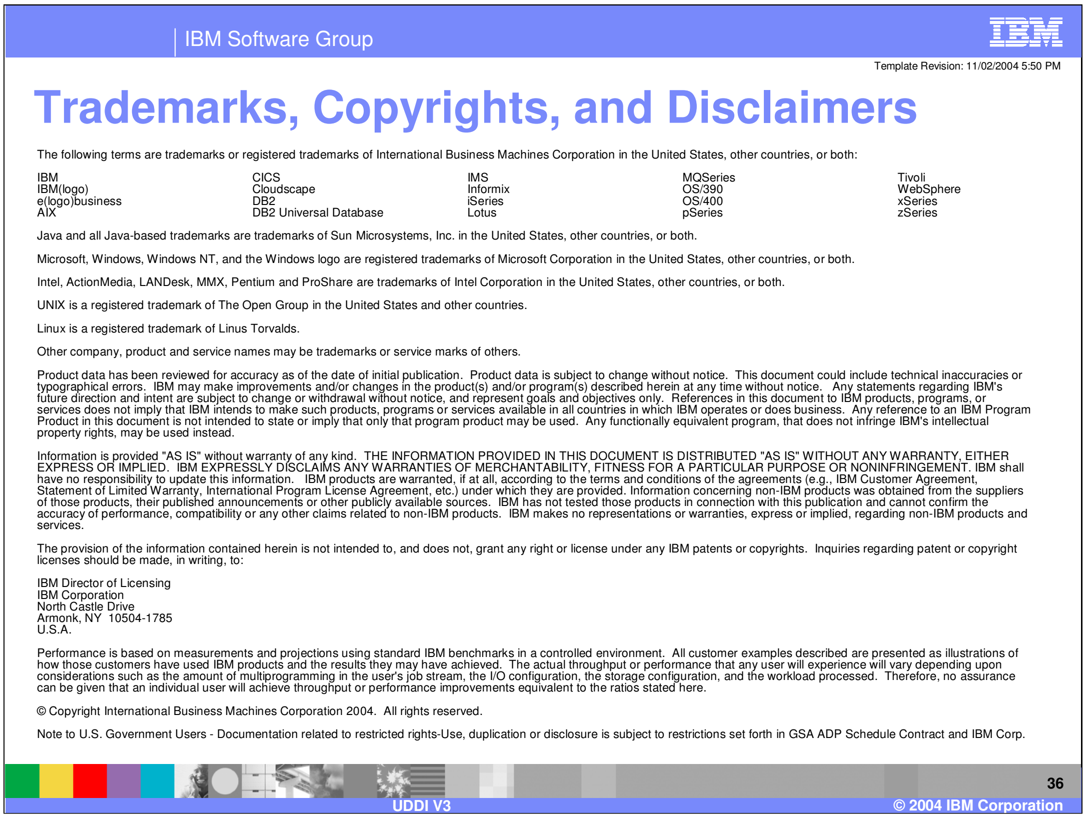
One reason for this apparently complicated migration path is that the format of the data has changed.



Summary

- The UDDI V3 specification provides significant enhancements over V2
  - UDDI Keys, Digital Signatures, Policy, Discovery, Multi-Registry
- WebSphere Application Server's UDDI V3 Registry can be managed from the Administrative console
- UDDI can be deployed at any time after installation
- Tools provided to migrate from UDDI V2 to V3
- UDDI V3 specification available at:

  http://uddi.org/pubs/uddi_V3.htm



Template Revision: 11/02/2004 5:50 PM

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
IBM(logo)
e(logo)business
AIX
CICS
Cloudscape
DB2
DB2 Universal Database
IMS
Informix
iSeries
Lotus
MQSeries
OS/390
OS/400
pSeries
Tivoli
WebSphere
xSeries
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.