ICE: Information and Content Exchange Protocol

Full ICE Specification

Version 2.0

2004 08 01

This version
http://www.icestandard.org/Spec/SPEC-ICE-2.0Full.pdf
Latest version
http://www.icestandard.org/Spec/SPEC-ICE2.0d.pdf
Previous version
http://www.icestandard.org/Spec/SPEC-ICE1.1.htm

Editors:
Jay Brodsky, Tribune Media Services
Marco Carrer, Oracle Corporation
Bruce Hunt, Adobe Systems, Inc.
Dianne Kennedy, IDEAlliance
Daniel Koger, Independent Consultant
Richard Martin, Active Data Exchange
Laird Popkin, Warner Music Group
Adam Souzis, Independent Consultant

Status of this Document

This document is an approved IDEAlliance Specification. It represents a significant step towards a stable specification suitable for widespread dissemination and implementation. It has been reviewed and approved by the ICE Authoring Group of IDEAlliance.

ICE 2.0 is the first major revision of the ICE Specification. As such, ICE 2.0 is not a compatible update to the ICE 1.0 specification. This update is a response to the implementation experience that has been gained over the past four years as well as the advancement in technology and W3C Recommendations. It differs from the ICE 1.0 and ICE 1.1 specifications in that it is specifically designed to support a Web Services model for syndication, has been modularized, incorporates XML Namespaces, and moves from an XML DTD to XML Schema.

As of this publication, the ICE Specification has been organized into a set of documents. This is one document in a set of documents (ICE Primer: Introduction and Overview, ICE Cookbook, Basic ICE Specification , Full ICE Specification, ICE Schemas and Scripts, and Guidelines to Extending the ICE Protocol) intended to jointly replace ICE 1.1. It has been developed by the IDEAlliance ICE Authoring Group. New documents may be added to this set over time.

The ICE Authoring Group and IDEAlliance recommend that implementations be updated to conform to the new ICE 2.0 Specification. The new specification embraces the latest Web technologies and W3C Recommendations. It provides added functionality that greatly enhances the usability of the protocol in a very wide range of syndication applications and can provide a substantial foundation for delivering syndication solutions in a Web Services environment.

Abstract

This document describes the Information and Content Exchange protocol for use by content syndicators and their subscribers. The ICE protocol defines the roles and responsibilities of Syndicators and Subscribers, defines the format and method of content exchange, and provides support for management and control of syndication relationships. We expect ICE to be useful in automating content exchange and reuse, both in traditional publishing contexts and in business-to-business relationships where the exchange eBusiness content must be reliably automated.

Table of Contents

1. Full ICE Overview

Basic ICE provides for a very simple syndication model where the Subscriber does not have an ICE server running constantly and polls for content as required. But, when more robust syndication functionality is required, *Full ICE* is appropriate. Full ICE extends Basic ICE functionality to add subscription management services as well as other advanced capabilities such as “push” delivery.

Full ICE also differs from Basic ICE in that the Subscriber is a sophisticated server implementation capable of not only sending ICE requests, but also receiving communications initiated by the Syndicator, such as the "push" of new content. In a Full ICE implementation both the Syndicator and Subscriber have an ICE server running at all times. Each must support SOAP transport bindings as well as subscription management capabilities.

Because Full ICE is an extension of Basic ICE, a Basic ICE implementation can talk to a Full ICE implementation, but without the advantages of Full ICE.

2. A Full ICE Scenario

Let's look at a step-by-step example of a simple transaction between a Syndicator and a Subscriber in a familiar industry. The Syndicator, the Best Code Company, a software developer, sets up and delivers a subscription to Tech News, a trade journal for the high technology industry. See Figure 2.1.

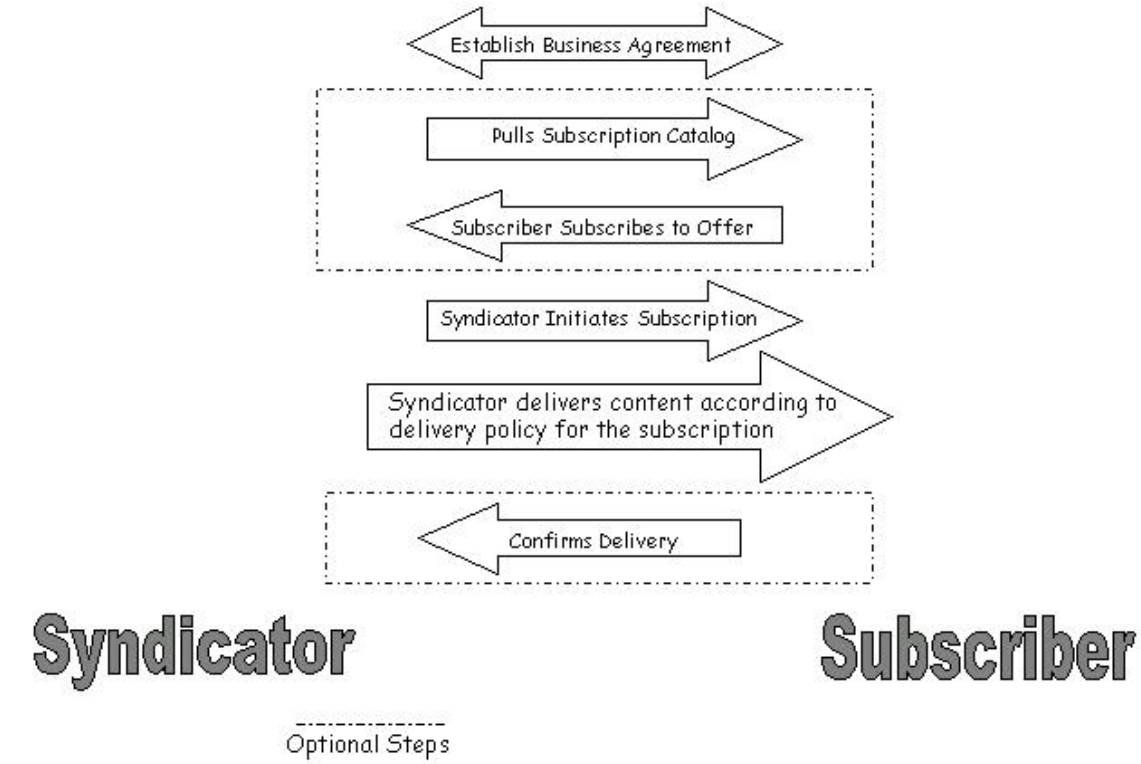

Figure 2.1 Full ICE Scenario

2.1 Syndicator and Subscriber Set up a Business Agreement

Syndication relationships begin with a business agreement. Best Code and Tech News agree on such terms as payment issues, usage rights, and subscription lifetime. The business agreement negotiation happens *outside* ICE and can involve person-to-person discussion, legal review, and contracts. Alternatively, a Syndicator could standardize and automate these terms.

2.2 Syndicator and Subscriber Set up a Subscription

Once the business agreement is in place, ICE comes into play as Best Code and Tech News start exchanging ICE messages to establish a subscription and begin content delivery.

2.2.1 Subscriber Receives Packages of Subscription Offers

In order to view a catalog of subscription offers, Tech News goes to the website of Best Code where the ICE Syndicator's end point is listed. The Subscriber may also use the discovery mechanism of *Universal Description, Discovery, and Integration (UDDI)* to find the Syndicator's end point. UDDI represents a set of protocols and a public directory for the registration and lookup of web services specified by UDDI.org. The Subscriber then requests a package of subscription offers using `<icedel:get-packages><icedel:get-package`. By convention, the subscription with the `subscription-id="1"` returns a Syndicator's catalog of subscription offers.

2.2.2 Subscriber Sends a Request to Subscribe to the Offer

Tech News thinks the press releases are exciting stuff and promptly asks to sign up for the subscription offer. It agrees to pull the content from Best Code's site.

2.2.3 Syndicator Accepts Request and Responds with Subscription Message

Best Code indicates that it has issued a subscription for Tech News by returning the `<icesub:subscription` message. Best Code gives Tech News a unique subscription-id number and also returns the details of the offer in order to confirm the delivery method.

2.3 Subscriber Receives Content

Once the subscription is set up, Tech News is ready to receive content. Tech News starts by asking for new content. Best Code has chosen to take advantage of the Full ICE incremental update capability. This means that the updates contain only changes to the content in the subscription. These changes can add new content and can also include requests to remove outdated content. In this way, Best Code can control the precise

content for that subscription on the Tech News site. Together with the actual content, the messages may also specify other subscription parameters such as effective date and expiration date.

2.3.1 Subscriber Requests Initial Subscription Content

Tech News uses `<icedel:get-package` to ask for subscription content. The `current-state="ICE-INITIAL"` indicates that this is an initial request for this subscription, which alerts Best Code to download the full content.

2.3.2 Syndicator Responds with Full Content of Subscription

Now Best Code delivers the content of its subscription, consisting of an ICE package with a press release and a video file. The press release is part of the package. It is the content of an ICE item element. The video file, however, is not actually in the package. Instead, its location is given in the URL attribute of an ICE `<icedel:item-ref` element. This serves as a pointer to the content and is an alternative to sending the content within the ICE message.
The ICE package element also conveys other information. For example, `editable="true"` gives Tech News permission to edit the content, while `new-state="2"` establishes the state of the subscription. The next time Tech News requests content, it will receive only content added or changed since this delivery, instead of receiving the entire content load all over again.

2.3.3 Subscriber Confirms Delivery

If requested by the Syndicator, the Subscriber will return confirmation of content delivery each time content is updated with the `<icedel:package-confirmations` message.

2.3.4 Variations on the Full ICE Scenario

Figure 2.1 shows the Full ICE subscription model. Note that an ICE subscription always begins with an out-of-bank business agreement between the Syndicator and the Subscriber. Several steps within the subscription model are optional, depending upon the business agreement. See steps marked with dotted lines in Figure 2.1. For example, the subscription might be initiated without the use of a catalog of offers. In this case the Syndicator can simply issues an `<icesub:subscription` message, provide a unique subscription identifier and begin delivering content.

3. Transport and Messaging

Two entities are involved in ICE transport and messaging. The *Syndicator* produces content that is delivered to *Subscribers.* The philosophy behind Full ICE is to enable syndication as a Web service. A Full ICE implementation implements all the features of the ICE 2.0 specification and supports SOAP transport bindings. Full ICE requiresthat the Syndicator and the Subscriber will establish a "listener" to receive messages and that either side will be able to request/respond and send/receive.

3.1 SOAP Binding with SOAP Request Respond Message Pattern

Because Full ICE supports SOAP transport bindings, two WSDL scripts are included in the ICE 2.0 Specification. These scripts define the transport for the Full ICE Syndicator and for the Full ICE Subscriber. The WSDL scripts can be found, in their entirety in *ICE: Schemas and Scripts*.

```
<!--  SOAP Binding  -->
  <binding name="ice-syndicator-full-binding"
 type="tns:ice-syndicator-full-portType">
  <soap:binding style="document"
  transport="http://schemas.xmlsoap.org/soap/http"/>
<operation>
   OPERATIONS GO HERE
</operation>. . .
  </binding>
```

3.2 Integrated ICE/SOAP Message

As was discussed in *ICE 2.0 Primer*, ICE was specifically designed to function as a Web service and to take advantage of SOAP as a messaging protocol. The ICE message header was designed to be carried within the SOAP header and the ICE delivery and subscription mechanisms were designed to be enclosed in the SOAP body.

4. Subscription Management

Basic ICE does not support subscription management. Only with Full ICE conformance may a Syndicator manage the subscriptions and data feeds to individual subscribers. Of course, Full ICE Syndicators may also provide public syndication feeds that are freely available to all Subscribers. But Full ICE was designed to support the business management of content syndication.

4.1 Subscription Establishment Overview

Subscription relationships in ICE usually begin with a request by the Subscriber to obtain a *catalog* of *subscription offers* from the Syndicator. As already described, prior to the Subscriber making this request, the Subscriber and the Syndicator have already engaged in discussions regarding licensing terms, payment options, and other business considerations. This happens outside of the ICE protocol. Once the parties agree that they wish to have a content exchange relationship, the ICE process begins.
A typical sequence of events is:

1. A user (technical manager, engineer, etc.) at the Syndicator site creates a new Subscriber account using the ICE software on the Syndicator's system. This operation is not defined by the protocol; it is a property of the tools used by the Syndicator.
2. The Syndicator tells the Subscriber what URL to use for ICE communication. It is likely that this URL will be under access control, and the Syndicator will communicate the necessary authentication data to the Subscriber using an out-of-band mechanism.
3. ICE protocol operations are now ready to begin: the Subscriber will authenticate (if necessary) to the given URL and issue the first ICE request: a `<icedel:get-packages><icedel:get-package subscription-id="1">` request for the package containing the catalog of offers.
4. The Syndicator will return a package containing offers.
5. The Subscriber issues a subscription request for an offer using the `<icesub:subscribe` message
6. The Syndicator responds with the `<icesub:subscription` message indicating that the subscription is established and packages can begin to be exchanged.

4.2 Get Package of Offers

The first step in the establishment of a subscription is the request from the Subscriber for a package containing a package of offers. This request is initiated by the `<icedel:get-package` message with the `subscription-id="1"` that is universally known as a

package that contains offers available from the Syndicator. See Figure 4.1. Change to get-package as the root!

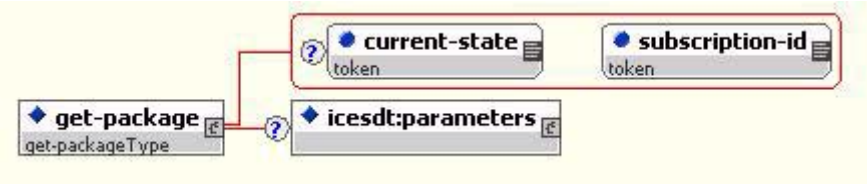

Figure 4.1 Get-package Request Structure

An example of such a request is shown below in its complete ICE/SOAP form:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env=”http://www.w3.org/2002/12/soap-
envelope”>
  <env:Header>
<icemes:Header
  xmlns:icedel=http://icestandard.org/ICE/V20/message
  timestamp=”2003-03-03T00:00:00” message-id=”m0056”>
  <icemes:sender name=”mycompany”
role=”http://icestandard.org//role/subscriber”
sender-id=”http://www.xxyz.org”/>
</icemes:Header>
  </env:Header>
  <env:Body>
 <icedel:get-package
 xmlns:icedel="http://icestandard.org/ICE/V20/delivery"
  subscription-id="1"/>
  </env:Body>
<env:Envelope>
```

4.3 Offers

The structure of an offer is shown in Figure 4.2. There are numerous attributes on offer. It is made up of optional `<icesub:content-metadata`, `<icesub:offer-metadata`, `<icesub:description`, followed by a required `<icesub:delivery-policy>` that can be followed by an optional `<icesub:business-term, one or more <icesub:required-extension` and other content from the Syndicator’s own schemas (#wildCard).

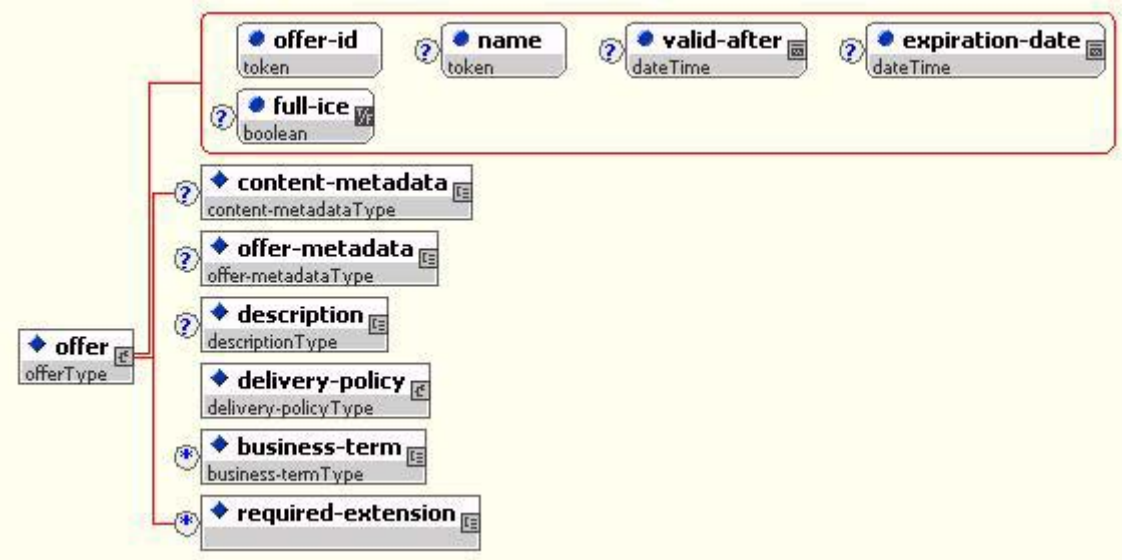

Figure 4.2 ICE Offer Structure

The element `<icesub:offer` is defined as the complex type *offerType* as shown in the following fragment of the XML schema:

```
<xs:element name = "offer" type = "offerType"/>
<xs:complexType name = "offerType">
  <xs:sequence>
   <xs:element name = "content-metadata"
  type = "content-metadataType" minOccurs = "0"/>
   <xs:element name = "offer-metadata"
  type = "offer-metadataType" minOccurs = "0"/>
   <xs:element name = "description"
  type = "descriptionType" minOccurs = "0"/>
   <xs:element name = "delivery-policy"
  type = "delivery-policyType"/>
   <xs:element name = "business-term"
  type = "business-termType" minOccurs = "0"
  maxOccurs = "unbounded"/>
   <xs:element name = "required-extension"
  minOccurs = "0" maxOccurs = "unbounded">
<xs:complexType>
 <xs:complexContent>
  <xs:extension base = "required-extensionType">
   <xs:attribute name = "extension-type"
  use = "required" type = "xs:anyURI"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name = "offer-id" use = "required"
 type = "xs:token"/>
  <xs:attribute name = "name" type = "xs:token"/>
  <xs:attribute name = "valid-after" type = "xs:dateTime"/>
```

```
  <xs:attribute name = "expiration-date"
 type = "xs:dateTime"/>
  <xs:attribute name = "full-ice" default = "false"
 type = "xs:boolean"/>
  <xs:anyAttribute namespace = "##other"
 processContents = "lax"/>
</xs:complexType>
```

4.4 Offer Attributes

An `<icesub:offer` has the following attributes:

- **offer-id**
  Required. This is a string. It is an identifier thatMUST be unique across all catalog offers between a Syndicator and Subscriber. Its function is to clearly identify this offer from all other catalog offers made by a Syndicator to a Subscriber.
- **name**
  Optional. This is a string. It is a name that may be used to distinguish subscriptions and offers from other subscriptions or offers. This is provided for use by the syndicator or subscriber and has no defined ICE semantics. Its intended use is to provide a readable short description of the offer such as, "Julia Child's Contemporary French Cooking Column".
- **full-ice**
  Default. This is Boolean. The default is set to ”false” for Basic ICE.
- **valid-after**
  Optional. This attribute has a dateType datatype. It is used to specify a date when the offer becomes valid and may be accepted.
- **expiration-date**
  Optional. This attribute has a dateType datatype. It is used to specify a date when the offer expires and is no longer valid.

4.5 Offer Elements

An offer is made up of a optional elements `<icesub:content-metadata`, `<icesub:offer-metadata, <icesub:description`, `<icesub:delivery-policy` `<icesub:business-term` , `<icesub:required-extensions` and allows for the inclusion of content from the Syndicator’s own schema (`#wildCard)`.

4.5.1 Content Metadata

Content-metadata is an element that provides the means for additional metadata that applies to all of the content being offered. The structure of this element is shown in Figure 4.3

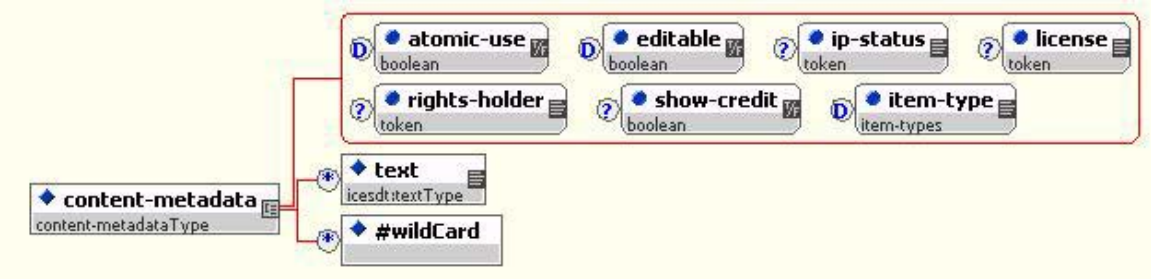

Figure 4.3 Content-metadata Structure

The `<icesub:content-metadata` element provides a mechanism to include additional metadata about the content. The metadata can be entered in the text field or other content metadata from the Syndicator's own schemas (`#wildCard`) can be included. Note that the `xml:lang=` attribute enables the specification of language used in the `<icesub:text` element.In addition, a number of optional metadata fields are provided as content-metadata attributes by the ICE 2.0 specification.
The attributes of `<icesub:content-metadata` include:

- **atomic-use**
  Optional. This is a Boolean. If "`true`", indicates that all information in the subscription must be used together, or not used at all. If "`false`", or unspecified, then the Subscriber is permitted to use subsets of the data in any way they want (and as permitted by the licensing terms, of course). This flag is meant to be useful as a hint/reminder displayed in a Subscribers ICE tool; ICE cannot enforce it (and, the use of lower case "must" in the above description is intentional; there is no protocol requirement here).
- **ip-status**
  Optional. This is a string describing the intellectual property-rights status of the content. ICE cannot enforce any of these semantics; rather, the intent is that this attribute allows the Syndicator to communicate useful information to the Subscriber ICE tool, which will ideally display this information in some useful presentation form. This attribute **MAY** contain any arbitrary string determined by the Syndicator. ICE defines the following specific string values, and Syndicators **SHOULD** use them as appropriate:

  - `PUBLIC-DOMAIN`
    The content has no licensing restrictions, whatsoever.
  - `FREE-WITH-ACK`
    The content has no licensing restrictions beyond a requirement to display an acknowledgement of the content source.
  - `SEE-LICENSE`
    The content has licensing restrictions as already agreed to in an existing licensing agreement. This is meant to convey the default case.
  - `SEVERE-RESTRICTIONS`
    The content has licensing restrictions that are worthy of special attention. NOTE: it is the intent that this flag would not be used routinely by Syndicators. The intent is that an ICE tool might "red flag" content

marked with this attribute and bring it specially to the attention of an administrator on the Subscriber site (this makes more sense when this attribute is attached to package items).

  - `CONFIDENTIAL`
    The content is confidential and must be protected specially.

- **license**
  Optional. Token indicating the license for the content.
- **rights-holder**
  Optional. String describing the original source of the syndication rights.
- **show-credit**
  Optional. This is a Boolean. If `true`, indicates that the Subscriber is explicitly expected to acknowledge the source of the data.
- **editable**
  Optional. This is a Boolean. If `true`, indicates that the Subscriber may edit/alter the content before using it. If `false`, or unspecified, the Subscriber is expected to use the content without any alteration. It has the same "hint" semantics as *atomic-use*.
- **item-type**
  Optional. This attribute is used to specify the type of content item that is being offered. The datatype is a URI that specifies the content type. This attribute was designed to indicate the datatype of the content of the subscription so that the subscriber will know whether they can process the content of the subscription being offered.

This is an example of `<icesub:content-metadata`:

```
<icesub:content-metadata
  atomic-use="true"
  editable="false"
  ip-status="Free With Acknoledgement"
  rights-holder="Oracle Corporation, 2003"
  show-credit="true"
  item-type="http://icestandard.org/ICE/V20/item-
type/rss2.0"/>
```

NOTE: The item-type attribute in this example is used to specify the “flavor” of RSS being used in the content.

4.5.2 Offer Metadata

Offer-metadata is an element that provides the means for additional metadata to be communicated between the parties specific to an offer. The structure of this element is shown in Figure 4.4.

Figure 4.4 Offer Metadata Structure

The `<icesub:offer-metadata` element provides a mechanism to include additional metadata about the offer. The metadata can be entered in the text field or other content metadata from the Syndicator's own schemas (#wildCard) can be included.

4.5.3 Description

This element is a text field and facilitates the entry of a description of the offer. This simple element is shown in Figure 4.5. Note that the `xml:lang=` attribute on `<icesub:text` enables the specification of language for the text field within `<icesub:description`.

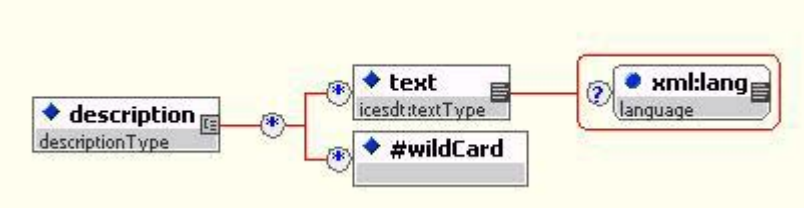

Figure 4.5 Description Element Structure

4.5.4 Delivery Policy

Each subscription offer has one delivery-policy. The delivery policy can determine, for example, the times and dates during which packages can be delivered (push) or pulled for a given subscription. Each delivery policy has one or more delivery rules.

The subscriber must accept the delivery policy within an offer and all (`required="true"`) delivery rules within a delivery policy. They can select among optional(`required="false"`) delivery rules, however.

See Figure 4.6 for the delivery-policy structure.

Figure 4.6 Delivery-policy Structure

The `<icesub:delivery-policy` element is defined as the type *delivery-policyType*. This is defined by the following XML schema fragment:

```
<xs:complexType name = "delivery-policyType">
  <xs:sequence>
<xs:element name = "delivery-rule"
  type = "delivery-ruleType" maxOccurs = "unbounded"/>
<xs:any namespace = "##other"
  processContents = "lax" minOccurs = "0"
  maxOccurs = "unbounded"/>
  </xs:sequence>
  <xs:attribute name = “startdate” type =
“icestd:dateTime”/>
  <xs:attribute name = “stopdate” type =
“icestd:dateTime”/>
  <xs:attribute name = "quantity" type = "xs:integer"/>
  <xs:attribute name = "expiration-priority"
 default = "first">
   <xs:simpleType>
<xs:restriction base = "xs:NMTOKEN">
  <xs:enumeration value = "first"/>
  <xs:enumeration value = "time"/>
  <xs:enumeration value = "quantity"/>
  <xs:enumeration value = "last"/>
  </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:anyAttribute namespace = "##other"
  processContents = "lax"/>
</xs:complexType>
```

The attributes for `<icesub:delivery-policy` are:

- **startdate**
  Optional. Datatype is `icesdt:dateTime` as defined in the ICE datatype schema. This attribute specifies the date and time on which the delivery schedule will start to apply. If this attribute is omitted, the schedule will start immediately.
- **stopdate**
  Optional. Datatype is `icesdt:dateTime` as defined in the ICE datatype schema. This attribute specifies the date and time on which the delivery schedule expires. If this attribute is omitted, the schedule never expires (unless superseded in the future).
- **quantity**
  Optional. Datatype is an integer. This attribute specifies the quantity of updates in the subscription
- **expiration-priority**
  Default. This attribute specifies the expiration priority. Values are `first, last, time`, and `quantity`. If the value is "`first`", then the subscription terminates when the first of the quantity or the expiration date is reached. If the value is "`last`", then the subscription terminates when both the quantity and the expiration date are reached. If the value is "`time`", then the subscription

terminates when the expiration date is reached. If the value is "`quantity`" then the subscription terminates when the quantity is reached. Note that expiration-priority has no effect unless both expiration-date and quantity are specified. The default is `first`.

> NOTE: The multiple delivery-rules in a delivery-policy are conceptually joined with "OR" (not "AND"). In other words, the valid delivery times are the union of all the times defined by each rule in the delivery policy.

4.5.4.1 Delivery Rule

Each `<icesub:delivery-policy` is made up of one or more delivery rules. The `<icesub:delivery-rule` can define a window of time during which deliveries can be performed along with other delivery options. Each delivery-rule has a mode of either a push or pull, can define when deliveries can be performed, a start and ending time for the update window, the frequency with which updates can be performed, the count of the number of updates that can be performed and the transport and packaging. In addition, attributes on the delivery rule specify whether updates will be full or incremental, whether the delivery of updates must be confirmed and whether this delivery rule is required. You can see the makeup of a `<icesub:delivery-rule` in the Figure 4.7:

Figure 4.7 ICE Delivery-rule Structure

The `<icesub:delivery-rule` element is defined as the type *delivery-ruleType* and is described by this XML Schema fragment:

```
<xs:complexType name = "delivery-ruleType">
  <xs:sequence>
  <xs:element name = "transport"
  maxOccurs = "unbounded" minOccurs = "1"
  type = "transportType"/>
<xs:any namespace = "##local ##other"
processContents = "lax" minOccurs = "0"
maxOccurs = "unbounded"/>
  </xs:sequence>
  <xs:attribute name = "mode" default = "pull">
   <xs:simpleType>
<xs:restriction base = "xs:NMTOKEN">
  <xs:enumeration value = "pull"/>
  <xs:enumeration value = "push"/>
</xs:restriction>
   </xs:simpleType>
```

```
  </xs:attribute>
  <xs:attribute name = "monthday" type = "xs:NMTOKENS"/>
  <xs:attribute name = "weekday" type = "xs:NMTOKENS"/>
  <xs:attribute name = "starttime" type = "icesdt:time"/>
  <xs:attribute name = "duration" type =
"icesdt:duration"/>
  <xs:attribute name = "min-num-updates"
type = "xs:integer"/>
  <xs:attribute name = "max-num-updates"
type = "xs:integer"/>
  <xs:attribute name = "incremental-update"
type = "xs:boolean" default = "false"/>
  <xs:attribute name = "required" type = "xs:boolean"
default = "true"/>
  <xs:attribute name = "confirmation" default = "false"
type = "xs:boolean"/>
  <xs:anyAttribute namespace = "##other"
processContents = "lax"/>
</xs:complexType>
```

Attributes on `<icesub:delivery-rule` include:

- **mode**
  Default. This attribute specifies the mode for the delivery. Options are `push` from Syndicator to Subscriber and `pull` by Subscriber from Syndicator, with a default of "`pull`" to support Basic ICE.
- **incremental-update**
  Default. This attribute specifies the update policy for the offer. The values are Boolean with "false" as the update default.
- **Confirmation**
  Default. This attribute specifies whether the Subscriber must confirm delivery of updates.The values are Boolean with "false" as the update default.
- **required**
  Default. This attribute specifies whether this delivery rule is required in order for the offer to be accepted. The values are Boolean with the default as "true". For example if a Syndicator provides a variety of delivery rules for the Subscriber to pick from, these rules would be optional.
- **weekday**
  Optional. This token indicates the day of the week on which delivery is scheduled .
- **monthday**
  Optional. This token indicates the day of the month on which delivery is scheduled.
- **duration**
  Optional. Datatype `icesdt:duration` as defined in the ICE datatype schema. This attribute specifies the duration of the window that starts at start-time everyday. .

- **min-num-updates**
  Optional. This attribute specifies the minimum number of updates. The datatype is an integer.
- **max-num-updates**
  Optional. This attribute specifies the maximum number of updates. The datatype is an integer.

4.5.4.1.1 Transport

The `<icesub:delivery-rule` is made up of one or more `<icesub:transport`. Transports are specified when the Syndicator makes an offer. This element provides a mechanism for the Syndicator to indicate the possible delivery transports for the `<icesub:offer`. You can see the makeup of a `<icesub:transport` in the Figure 4.8.

Figure 4.8 Transport Structure

The `<icesub: transport has two attributes:`

- **protocol**
  Default. This attribute specifies the transport protocol. It has pre-enumerated values of "`http:get`" "`ftp`" "`mailto`" and "`soap`" with the default set to "`http:get`" for Basic ICE.
- **packaging-style**
  Default. This attribute specifies the packaging style for the offer. It has pre-enumerated values of "`ice`" and "`raw`" with the default set to "`ice`" for Basic ICE.

4.5.4.1.2 Delivery-Endpoint

The <icesub:transport is made up of an optional delivery endpoint. See Figure 4.9. If the Syndicator is offering content in "`pull`" mode, the delivery endpoint can be specified by using `<icesub:delivery-endpoint`. If the Syndicator is offering content in "`push`" mode, the Subscriber would use this elements within the `<icesub:subscribe` message to indicate the endpoint for the push delivery.

Figure 4.9 Delivery-endpoint Structure

The <icesub:delivery-endpoint has 4 attributes. These include:

- **url**
  Required. This attribute specifies the URL for push delivery. The datatype is anyURI.
- **username**
  Optional. This attribute specifies an optional username that may be required to access URL for push delivery.
- **password**
  Optional. This attribute specifies an optional password that may be required to access URL for push delivery.
- **user-authentication**
  Optional. This attribute specifies the optional user authentication scheme. It is a string with enumerated values of "basic" and "digest". There is no default

4.5.4.2 Syndicator Offer Specifications by Mode

One of the most important specifications within the delivery rule of an offer is the specification of delivery mode= along with the <icesub:syndicator-transports. The requirement to specify syndicator transports delivery settings varies by delivery mode. Conditions such as this cannot be expressed by XSD. The following table provides required specifications based on delivery mode.

Mode	Syndicator Protocol	Syndicator Delivery Packaging Style	Delivery Endpoint
Pull	**Default**. If not specified, the default protocol will be "http:get"	**Default**. If not specified, Syndicator packaging is assumed to be "ice"	**Optional.** If not specified it is assumed to be the same endpoint from where the catalog was pulled
Push	**Required**. For push delivery a specific protocol should be indicated because http:get is not a push protocol	**Required**. For push delivery a specific protocol should be indicated	**Not Allowed.** For push delivery, only the Subscriber delivery endpoint is valid

4.5.4.3 Example Delivery Rules

In this section we will look at a number of offers with delivery rules within delivery policies. The intent is to provide examples of delivery rules with different modes and Syndicator specifications.

4.5.4.2.1 Simple "Pull" Delivery Rule

First lets look at an offer with a `<icesub:delivery-rule` containing a simple "`pull`". Notice that in this simple rule, everything is left to default including the mode on the delivery rule. No transport protocol or packaging-style is provided. Remember that the assumption is that the protocol will be "`http:get`" and the packaging will be "`ice`". The pull will be made from the location specified by the `<icesub:delivery-endpoint`.

```
<icedel:package
 xmlns:icedel="http://icestandard.org/ICE/V20/delivery"
 new-state="ICE-ANY"
 old-state="ICE-ANY"
 fullupdate="true"
 package-id="1"
 subscription-id="1">
  <icedel:add>
   <icedel:metadata item-
type="http://icestandard.org/ICE/V20/item-type/offer"
content-type="text/xml"/>
<icedel:item>
  <icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2"
name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
   <icesub:delivery-policy>
  <icesub:delivery-rule>
          <icesub:transport>
           <icesub:delivery-endpoint
             url="http://www.iceserver.com/gp/08292BC" >
          </icesub:transport>
         </icesub:delivery-rule>
   </icesub:delivery-policy>
  </icesub:offer>
</icedel:item>
  </icedel:add>
</icedel:package>
```

NOTE: You can tell this ICE package contains a catalog offer in several ways. First notice that the `subscription-id` on the package equals "`1`". This is the identifier of a subscription catalog. Also notice that the `<icedel:metadata` indicates the item type is offer. And finally the offer is inside this package.

4.5.4.2.2 "Pull" Delivery Rule with Syndicator Delivery Settings

In this example, the delivery rule specifies a pull delivery. But rather than using the defaults, this Syndicator is specifying transport. In this case the Syndicator provides a delivery endpoint for content to be pulled from. The Syndicator also indicates that the delivery packaging style will "`ice`".

```
<icedel:package
 xmlns:icedel="http://icestandard.org/ICE/V20/delivery"
 new-state="ICE-ANY"
 old-state="ICE-ANY"
 fullupdate="true"
 package-id="1"
 subscription-id="1">
  <icedel:add>
   <icedel:metadata item-
type="http://icestandard.org/ICE/V20/item-type/offer"
content-type="text/xml"/>
<icedel:item>
  <icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2"
name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
   <icesub:delivery-policy quantity="100"
expiration-priority="quantity">
  <icesub:delivery-rule mode="pull"/>
  <icesub:transport
protocol="http:get" packaging-style="ice">
<icesub:delivery-endpoint
  url="http://iceserver.com/gp/08292BC"/>
  </icesub:transport>
</icesub:delivery-rule>
   </icesub:delivery-policy>
  </icesub:offer>
</icedel:item>
  </icedel:add>
</icedel:package>
```

NOTE: The `<icesub:delivery-policy` indicates that this subscription provides for a quantity of 100 feeds. In this case, the subscription expires when the quantity has been filled. Also note that no times or durations are placed on this subscription.

4.5.4.2.3 Single "Push" Delivery Rule

Instead of specifying that delivery will be by "`pull`", the Syndicator may indicate a "`push`" delivery. In this case the Syndicator provides protocol and packaging information but does not provide `<icesub:delivery-endpoint` as push endpoints have to be provided by the Subscriber!

```
<icedel:package
 xmlns:icedel="http://icestandard.org/ICE/V20/delivery"
 new-state="ICE-ANY"
 old-state="ICE-ANY"
 fullupdate="true"
 package-id="1"
 subscription-id="1">
  <icedel:add>
   <icedel:metadata item-
type="http://icestandard.org/ICE/V20/item-type/offer"
content-type="text/xml"/>
<icedel:item>
  <icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2"
name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
   <icesub:delivery-policy quantity="100"
expiration-priority="quantity">
 <icesub:delivery-rule mode="push">
  <icesub:transport protocol="soap"
  packaging-style=”ice”/>
  <icesub:transport protocol=”soap”
  packaging-style=”raw”/>
  <icesub:transport protocol=”ftp”
  packaging-style=”ice”/>
  <icesub:transport protocol=”ftp”
  packaging-style=”raw”/>
 </icesub:delivery-rule>
   </icesub:delivery-policy>
  </icesub:offer>
</icedel:item>
  </icedel:add>
</icedel:package>
```

NOTE: In this example the Syndicator provided four transport protocols/packaging style options. The Subscriber can select a preferred transport protocol and packaging style pair when subscribing to this offer.

4.5.4.2.4 Combined "Pull" and "Push" Delivery Rule

A Syndicator may specify that delivery will be by "pull" and "push" delivery. In this case the Subscriber must be able to accept both delivery rules (which default to required) in order to subscribe to the offer. The Subscriber can, however, choose a preferred transport within each rule.

```
<icedel:package
 xmlns:icedel="http://icestandard.org/ICE/V20/delivery"
 new-state="ICE-ANY"
 old-state="ICE-ANY"
 fullupdate="true"
 package-id="1"
 subscription-id="1">
  <icedel:add>
   <icedel:metadata item-
type="http://icestandard.org/ICE/V20/item-type/offer"
content-type="text/xml"/>
   <icedel:item>
<icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2"
name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
  <icesub:delivery-policy quantity="100"
expiration-priority="quantity">
  <icesub:delivery-rule mode="pull">
   <icesub:transport protocol=”http:get”
  packaging-style=”ice”>
 <icesub:delivery-endpoint
url="http://iceserver.com/ice/08292BC82302427"/>
   </icesub:transport>
   <icesub:transport protocol=”http:get”
  packaging-style=”raw”>
 <icesub:delivery-endpoint
url="http://iceserver.com/raw/08292BC82302427"/>
   </icesub:transport>
 </icesub:delivery-rule>
 <icesub:delivery-rule mode="push">
  <icesub:transport>
   <icesub:transport protocol=”soap”
  packaging-style=”ice”/>
   <icesub:transport protocol=”mailto”
  packaging-style=”ice”/>
 </icesub:delivery-rule>
   </icesub:delivery-policy>
  </icesub:offer>
```

```
</icedel:item>
  </icedel:add>
</icedel:package>
```

NOTE: Because the "required" attribute on delivery rule is left to default to "true" all delivery rules within this delivery policy must be accepted in order for the Subscriber to subscribe to the offer.

4.5.5 Offer Business Term

Another component of the <icesub:offer is the optional <icesub:business-term element. Business terms provide the means for additional content and parameters to be communicated between the parties; both for specific subscriptions as well as for more general properties of the relationship. You can see the structure of this element in Figure 4.10:

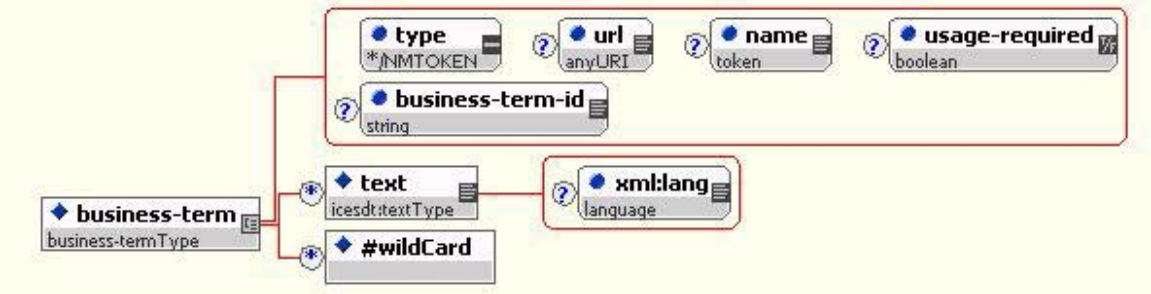

Figure 4.10 ICE Business-term Structure

The <icesub:business-term element is defined as the type *business-termType* and is described by this XML Schema fragment:

```
<xs:element name = "business-term" type = "business-
termType" minOccurs = "0" maxOccurs = "unbounded"/>
  <xs:complexType name = "business-termType" mixed =
"true">
   <xs:sequence>
<xs:element name = "text" type = "icesdt:textType"
minOccurs = "0" maxOccurs = "unbounded"/>
<xs:any namespace = "##other" processContents = "lax"
minOccurs = "0" maxOccurs = "unbounded"/>
   </xs:sequence>
   <xs:attribute name = "type" use = "required">
<xs:simpleType>
 <xs:restriction base = "xs:NMTOKEN">
 <xs:enumeration value = "credit"/>
 <xs:enumeration value = "licensing"/>
 <xs:enumeration value = "payment"/>
 <xs:enumeration value = "reporting"/>
</xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name = "url" type = "xs:anyURI"/>
  <xs:attribute name = "name" type = "xs:token"/>
```

```
  <xs:attribute name = "usage-required" type =
"xs:boolean"/>
  <xs:attribute name = "business-term-id" type =
"xs:string"/>
  <xs:anyAttribute namespace = "##other" processContents =
"lax"/>
</xs:complexType>
```

The attributes of `<icesub:business-term` are:

- **type**
  Required. String identifying the particular class of business terms. All of these terms are plain text descriptions. ICE makes no attempt to programmatically explain licensing agreements; rather, ICE simply provides a transport mechanism allowing user interfaces to easily locate, manage, and display electronic copies of license agreements presumably executed in the traditional way on paper. The type is one of the following values:

type attribute value	Attribute "type" Meaning
credit	Refers to the type of acknowledgement required when using the content. Note that ICE makes no further requirements about credit. The party making the offer (usually a syndicator) **MAY** choose to provide parameters in this category that MAY or MAY NOT be negotiable.
licensing	Refers to the general terms of licensing. Note that ICE makes no further requirements about licensing. The party making the offer (usually a syndicator) **MAY** choose to provide parameters in this category that MAY or MAY NOT be negotiable.
payment	Payment refers to the cost and payment terms expected when using the content. Note that ICE makes no further requirements about payment. The party making the offer (usually a syndicator) **MAY** choose to provide parameters in this category that MAY or MAY NOT be negotiable.
reporting	Refers to the end-user usage statistics expected when content is used. Note that ICE makes no further requirements about reporting (but see logging). The party making the offer (usually a syndicator) **MAY** choose to provide parameters in this category that MAY or MAY NOT be negotiable.

- **url**
  Optional. A url. URL has no protocol-defined semantics other than to be made available to the ICE application processor. The intent is that this URL provides the business terms.
- **name**
  Optional. This name MAY be used by an ICE application processor to identify

the specific business term. "name" has no protocol defined semantics other than to be made available to the ICE application processor.

- **usage-required**
  Optional. This attribute specifies whether the business term usage is required. This is a Boolean. If `true`, indicates usage is required and `false` indicates it is not required.
- **business-term-id**
  Optional. A subscription unique business term identifier that ICE uses to distinguish the business term from all other business terms in the subscription.

4.6 Subscribing

A Subscriber uses the `<icesub:subscribe` containing an `<icesub:offer` to establish a subscription. Typically, a Subscriber will use `<icedel:get-packages/<icedel:get-package` to get an initial package of offers, take one of the offers from that catalog of offers and send it back to the Syndicator in an `<icesub:subscribe` request.

4.6.1 Subscribe Element

The subscribe message is made up of an offer with parameters. See Figure 4.11.

Figure 4.11 The structure of the subscribe message

The `<icesub:subscribe` message can carry only a single offer. This means that there is a single offer per subscription. The `<icesub:offer` element is described in *4.3 Offers*.

In addition to the offer, the `<icesub:subscribe` message may contain the `<icesub:parameters` element. This element enables the Subscriber to send further parameters at subscription time to specify parameters. These parameters are not defined within an ICE 2.0 namespace, but rather must come from a Subscriber namespace.

The structure of the <icesub:subscribe message is shown in the XML schema fragment:

```
<xs:element name = "subscribe">
  <xs:complexType>
<xs:sequence>
 <xs:element name = "offer" type = "offerType" minOccurs =
"0"/>
 <xs:element ref = "icesdt:parameters" minOccurs = "0"/>
</xs:sequence>
<xs:attribute name = "subscription-name" type =
"xs:token"/>
<xs:attribute name = "offer-id" type = "xs:token"/>
<xs:anyAttribute namespace = "##other" processContents =
"lax"/>
  </xs:complexType>
</xs:element>
```

The <icesub:subscribe message has a two attributes:

- **subscription-name**
  Optional. This attribute specifies the name of the product being subscribed to.
- **offer-id**
  Optional. This attribute specifies the id of the offer being subscribed to. If this attribute is used, without an echo of the <icesub:offer, it means that the offer was accepted just as it was presented.

4.6.1.1 Subscribing Directly to an Offer

If an <icesub:subscribe is returned with the offer-id attribute but without an echo of the <icesub:offer, it means that the offer was accepted and subscribed to just as it was presented. The offer-id attribute was put on <icesub:subscribe specifically to allow for this short cut.

See how this is done in the following example:

```
<icesub:subscribe subscription-name="RSS Headlines"
 offer-id="offID2"/>
```

NOTE: The offer-id can only be used to subscribe to offers that are “pull” only. If an offer has “push” delivery rules, the Subscriber must return the offer with delivery endpoints for the push specified.

4.6.1.2 Subscribing with Subscriber Parameters Returned

If an offer has “push” delivery rules, the Subscriber must return the offer with delivery endpoints for the push specified. The Subscriber may also have been given choices of

delivery style selections that must be specified in order for content delivery to commence. In both these cases, the Subscriber must return the `<icesub:offer` within the `<icesub:subscribe`.

4.6.1.2.1 Subscriber Transport

The Subscriber returns transport for push deliveries back to the Syndicator within the offer that is returned in the `<icesub:subscribe` message. For information on `<icesub:transport`.

4.6.1.2.2 Example Subscribe Message with Subscriber Parameters

This example shows an `<icesub:subscribe` message in response to the offer shown in *4.5.4.2.4 Combined "Pull" and "Push" Delivery Rule.* In this example, the Subscriber sends the offer within the `<icesub:subscribe` message. Note that the Subscriber has selected one `<icesub:transport` option. Also `<icesub:delivery-endpoint` has been provided by the Subscriber so the Syndicator will know where the push delivery will be made.

```
<icesub:subscribe>
<icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2"
name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
  <icesub:delivery-policy quantity="100"
expiration-priority="quantity">
 <icesub:delivery-rule mode="push">
  <icesub:transport protocol="soap"
  packaging-style=”ice”>
<icesub:delivery-endpoint
  url="http://sub.com/push.jsp" username="foo"
  password="foofoo"/>
   </icesub:transport>
 </icesub:delivery-rule>
   </icesub:delivery-policy>
</icesub:offer>
</icesub:subscribe>
```

4.6.2 Subscription Initiated

After the Subscriber returns the offer to the Syndicator within a `<icesub:subscribe` message, the Syndicator can respond in one of two ways, depending upon whether the subscription was accepted.

If the Syndicator accepts the subscribe request, the Syndicator responds with the `<icesub:subscription` message shown in Figure 4.12.

Figure 4.12 Syndicator's Subscription Response

The structure of the subscription message can be seen in this XML schema fragment:

```
<xs:element name = "subscription" type =
 "subscriptionType"/>
  <xs:complexType name = "subscriptionType">
   <xs:sequence>
<xs:element name = "offer" type = "offerType"/>
 <xs:any namespace = "##other" processContents = "lax"
minOccurs = "0" maxOccurs = "unbounded"/>
   </xs:sequence>
   <xs:attribute name = "subscription-id" use = "required"
type = "xs:token"/>
   <xs:attribute name = "subscription-name" type =
"xs:token"/>
   <xs:attribute name = "current-state" type =
"icesdt:package-sequence-stateType"/>
   <xs:attribute name = "quantity-remaining" type =
"xs:integer"/>
   <xs:anyAttribute namespace = "##other" processContents =
"lax"/>
</xs:complexType>
```

The `<icesub:subscription` element has several attributes:

- **subscription-id**
  Required. This attribute specifies the unique identifier of the product being subscribed to. The Syndicator provides a subscription-id when the subscription begins.
- **subscription-name**
  Optional. This attribute specifies the name of the product being subscribed to.
- **current-state**
  Optional. This attribute specifies the current state of the subscription and is only used when the `<icesub:subscription` is returned within an `<ice:status` . It is datatype `icesdt:package-sequence-stateType` as defined in the ICE simpledatatypes.xsd. Values include `ICE-INITIAL` and `ICE-ANY`.
- **quantity-remaining**
  Optional. This attribute specifies the quantity of updates of the product being subscribed to and is only used when the `<icesub:subscription` is returned within an `<ice:status`. The datatype is an integer.

NOTE: The subscription does not have an offer-id as an attribute. This means that even though the offer-id can be used as a short-cut by the Subscriber when subscribing, the Syndicator is forced to repeat the entire offer within the subscription. This is a safeguard to ensure that the Subscriber clearly understands the subscription and all delivery policies at the time the subscription in initiated by the Syndicator.

An example of the `<icesub:subscription` message is shown below. This is the subscription that was established based on the example shown in *4.6.1.2.2 Example Subscribe Message with Subscriber Parameters.*

```
<icesub:subscription
   subscription-id="08292BC82302427F8CBC93342F931EC8"
   current-state="ICE-INITIAL"
   quantity-remaining="100">
<icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2" name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
  <icesub:delivery-policy quantity="100"
expiration-priority="quantity">
 <icesub:delivery-rule mode="push">
  <icesub:transport protocol="soap"
  packaging-style=”ice”>
<icesub:delivery-endpoint
  url="http://sub.com/push.jsp" username="foo"
  password="foofoo"/>
   </icesub:transport>
 </icesub:delivery-rule>
   </icesub:delivery-policy>
</icesub:offer>
</icesub:subscription>
```

Note The subscription message is returned directly within the SOAP body. Even though the original offer is sent inside a package, the subscription reply is not.

4.6.3 Subscription Declined

If the Syndicator declines the subscription, the response is `<icesub:subscription-fault`. The fault contains a fault code. One of the following fault codes is appropriate for declining a subscription.

- 400 Generic request error
  Generic status code indicating inability to comprehend the request. Usually, it is better to send a more specific code if possible.

- 401 Incomplete/cannot parse
  The request sent is severely garbled and cannot be parsed. Note that in most cases, a message level error (301) might be more appropriate.
- 402 Not well formed XML
  The request sent is recognizable as XML, but is not well formed per the definition of XML. This is available as both a message level error and as a request level (4xx) error. Whether a given implementation attempts to interpret not well formed XML so as to generate request level (4xx) errors versus. Message level (3xx) errors is a quality of implementation issue.
- 403 Validation failure
  *The request failed validation according to the Schema. This is available as both a message level error and as a request level (4xx) error. Whether a given implementation attempts to interpret not well formed XML so as to generate request level (4xx) errors versus. Message level (3xx) errors is a quality of implementation issue. Note that Receivers **SHOULD** perform validation on incoming ICE messages, but are not required to. Senders **MUST** send only valid ICE messages or they are in error; however, the ability to detect invalid messages is a quality-of-implementation issue for the Receiver, and Senders **MUST NOT** assume the Receiver will perform an XML validation on their messages.*
- 422 Schedule violation
  The subscribe request was made at an incorrect time such as after an offer has expired or before it is valid.
- 440 Sorry
  This indicates the Syndicator rejected the proposed subscription offer, but wishes to extend additional offers.

4.6.4 ICE Subscription Fault

The `<icesub:subscription-fault` is returned when a subscription is declined. The structure of the fault can be seen in Figure 4.13.

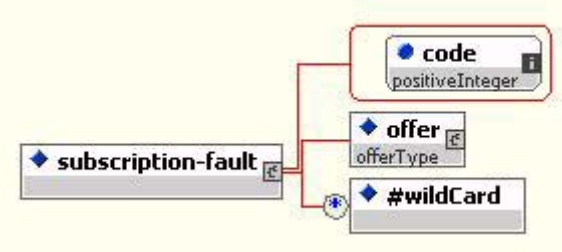

Figure 4.13 ICE Subscription Fault

The following is an example of a Syndicator declining a subscription:

```
<icesub:subscription-fault
   code="440">
<icesub:offer
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
offer-id="offID2" name="offName2">
  <icesub:description>
   headlines
  </icesub:description>
  <icesub:delivery-policy quantity="100"
expiration-priority="quantity">
 <icesub:delivery-rule mode="push">
  <icesub: transport protocol="soap"
  packaging-style=”ice”>
<icesub:delivery-endpoint
  url="http://sub.com/push.jsp" username="foo"
  password="foofoo"/>
   </icesub:transport>
 </icesub:delivery-rule>
   </icesub:delivery-policy>
</icesub:offer>

</icesub:subscription-fault>
```

5. Other Subscription Operations

In addition to providing the ability for the Subscriber to subscribe to an offer and for the Syndicator to approve and manage that subscription, Full ICE provides for two other important subscription management operations— checking the status of a subscription and cancellation of the subscription.

5.1 Get Status

ICE 2.0 provides the ability for the Subscriber to request the status of a subscription. The structure of the `<icesub:get-status` message is shown in Figure 5.1.

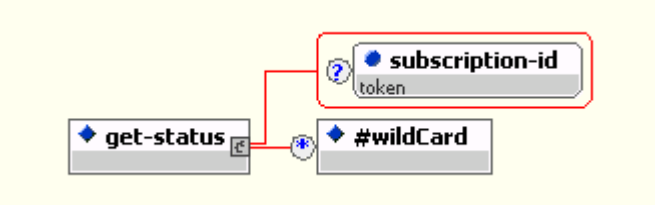

Figure 5.1 Get-status Structure

> **NOTE:** If the optional subscription-id is not provided, the Syndicator is expected to respond with the status of each subscription for the Subscriber.

The following example shows how the `<icesub:get-status` request is issued. Note that the entire ICE/SOAP message is shown.

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env='http://www.w3.org/2002/12/soap-
envelope'>
  <env:Header>
<icemes:Header timestamp=”2003-03-03” message-id=”m0056”>
  <icemes:Sender name=”mycompany”
 role=”http://icestandard.org//role/syndicator”
 sender-id=”http://www.xxyz.org”/>
  </icemes:Header>
  </env:Header>
  <env:Body>
<icesub:get-status subscription-id=”MC003”/>
  </env:Body>
<env:Envelope>
```

5.2 Status

ICE 2.0 provides the ability for the Syndicator to respond to the request from the Subscriber for the status of a subscription. The structure of the `<icesub:status` message is shown in Figure 5.2.

Figure 5.2 ICE Status Structure

The ICE Status response returns the subscriptionthat includes the current state of the subscription and the quantity remaining in the subscription and the subscription-id.. From this information, the Subscriber can answer any question that prompted the `<icesub:get-status` request.

5.3 Cancel

ICE 2.0 provides the ability for the Subscriber to cancel a subscription. The structure of the `<icesub:cancel` message is shown in Figure 5.3.

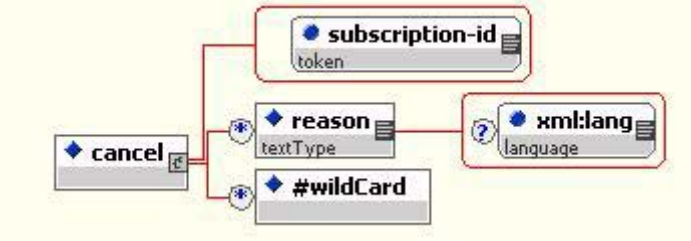

Figure 5.3 ICE Cancel Structure

The Subscriber's request to cancel a subscription simply includes the subscription-id for the subscription being cancelled and a reason attribute. The `xml:lang` attribute enables the Subscriber to specify the language for the reason text.
An example of an ICE cancel message is shown here:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env='http://www.w3.org/2002/12/soap-
envelope'>
  <env:Header>
<icemes:Header timestamp=”2003-03-03” message-id=”m0056”>
  <icemes:Sender name=”mycompany”
 role=”http://icestandard.org//role/syndicator”
 sender-id=”http://www.xxyz.org”/>
  </icemes:Header>
  </env:Header>
  <env:Body>
<icesub:cancel
  xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
  subscription-id="08292BC82302427F8CBC93342F931EC8">
  <icesub:reason xml:lang="en">
I'm tired of this content feed
  </icesub:reason>
</icesub:cancel>
</env:Body>
</env:Envelope>
```

5.4 Cancellation

ICE 2.0 provides the ability for the Syndicator to verify the cancellation of a subscription requested by the Subscriber with the `<icesub:cancel` message. The structure for the `<icesub:cancellation` response is shown in Figure 5.4.

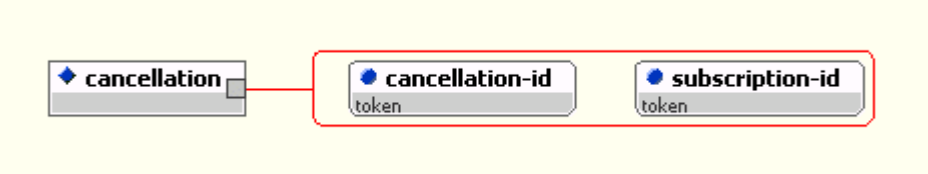

Figure 5.4 ICE Cancellation Response Structure

The Cancellation response requires the Syndicator to provide the Subscriber with a unique cancellation-id that can be used to verify the cancellation.
An example of an ICE cancellation response is shown here:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env='http://www.w3.org/2002/12/soap-
envelope'>
  <env:Header>
<icemes:Header timestamp=”2003-03-03” message-id=”m0056”>
  <icemes:Sender name=”mycompany”
 role=”http://icestandard.org//role/syndicator”
 sender-id=”http://www.xxyz.org”/>
  </icemes:Header>
  </env:Header>
  <env:Body>
<icesub:cancellation
xmlns:icesub="http://icestandard.org/ICE/V20/subscribe"
subscription-id="08292BC82302427F8CBC93342F931EC8"
cancellation-id="C08292BC82302427F8CBC93342F931EC8"/>
</env:Body>
</env:Envelope>
```

6. Packages and Delivery

Full ICE, like Basic ICE, supports the delivery of packages. In Basic ICE, the delivery is simply the act of the Syndicator placing the content in a SOAP/ICE XML document at the URL specified in the offer. Full ICE enables the push or pull of content. The XML definition for packages is found at http://www.icestandard.org/Spec/V20/schema/ice-delivery.xsd.

The Full ICE package is made up of three elements. See Figure 6.1

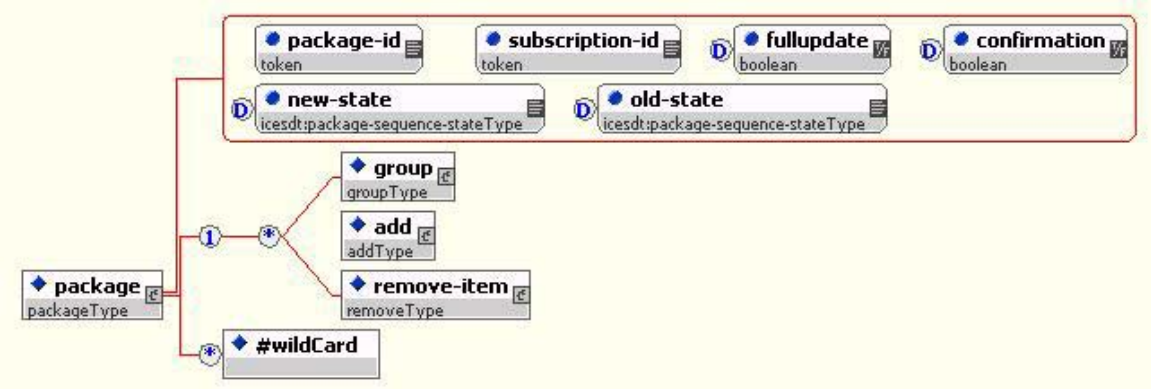

Figure 6.1 ICE Package Structure

The formal definition of a package is expressed with the following XML schema. Note the defaults of package attributes that define Basic ICE functionality.

```
<element name = "package" type = "icedel:packageType"/>
  <complexType name = "packageType">
<sequence>
  <group ref = "icedel:cm.package"/>
<any namespace = "##other" processContents = "lax"
minOccurs = "0"
 maxOccurs = "unbounded"/>
</sequence>
<attribute name = "package-id" use = "required" type =
"xs:token"/>
<attribute name = "subscription-id" use = "required" type =
"xs:token"/>
<attribute name = "fullupdate" default = "true" type =
"xs:boolean"/>
<attribute name = "confirmation" default = "false" type =
"xs:boolean"/>
<attribute name = "new-state" default = "ICE-ANY"
type = "icesdt:package-sequence-stateType"/>
 <attribute name = "old-state" default = "ICE-ANY"
type = "icesdt:package-sequence-stateType"/>
<anyAttribute namespace = "##other" processContents =
"lax"/>
  </complexType>
</element
```

6.1 Package Attributes

There are several attributes on `package`:

- **confirmation**
  Default. This attribute specifies whether confirmation of receipt is required. The values are Boolean, "true" or "false." The default is "false". Confirmation does not apply for Basic ICE since Basic ICE does not support subscription management.
- **fullupdate**
  Default. This attribute specifies whether the package contains a full (or partial) update. The values are Boolean, "true" or "false." The default is "true" because Basic ICE does not require management of incremental updates.
- **new-state**
  Default. One of two sequence identifiers, which, together represent the state of the subscription. Since Basic ICE does not support subscription management, the default is set to "ICE-ANY".
- **old-state**
  Default. One of two sequence identifiers, which, together represent the state of the subscription. Since Basic ICE does not support subscription management, the default is set to "ICE-ANY".
- **package-id**
  Required. Identifies the package within the scope of a subscription. It is referenced in certain `ice-code` messages such as 201 (Confirmed) and for package confirmations. The Syndicator assigns the `package-id`.
- **subscription-id**
  Required. In Basic ICE, the subscription-id is the unique id of the content feed and is used by all subscribers. The Syndicator assigns the `subscription-id`.

6.2 Package Elements

The ICE package is made up of 3 elements. See figure 6.2. An ICE `<icedel:package` describes a set of content operations: additions, removals, and a group of additions and/or removals that are used to update/distribute syndicated content. The content additions contain the content that needs to be added or updated and are specified using the `<icedel:item` and `<icedel:item-ref` elements. The `<icedel:group` element allows the Syndicator to associate the content specified using the `<icedel:item` elements together. For example, in the syndication of restaurant reviews, each review may consist of different types of content such as an HTML file and two graphic files. These three files could be contained within three `<icedel:item` elements and grouped together in an ICE `<icedel:group` as a single restaurant review. Likewise, unrelated content can be specified in a `<icedel:package` by just using the `add and then <icedel:item` elements without an intervening `<icedel:group`. The `<icedel:item` element is used to contain content directly for delivery. The `<icedel:item-ref` element is used to

distribute an indirect reference to the actual content. Note that the `#wildcard` allows for insertion of content from any namespace.

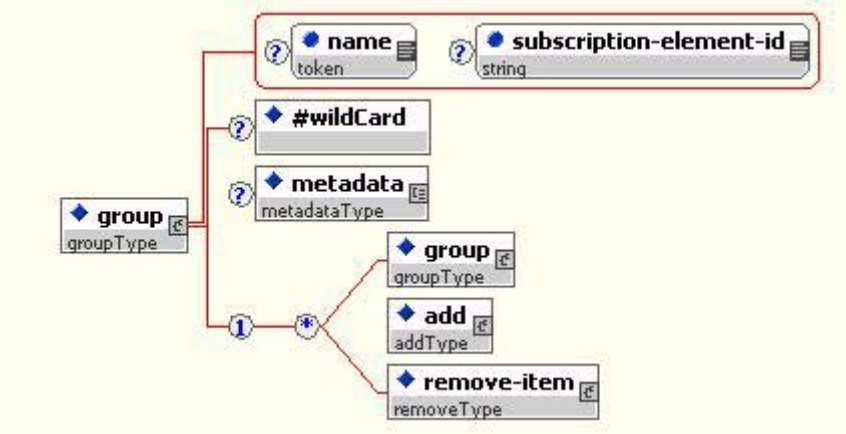

Figure 6.2 Package Elements for Full ICE

6.2.1 Group

The `<icedel:group` is a container element that can be used to group content items being added or removed. It also enables the attachment of metadata to a group of content items.

Attributes on `<icedel:group` include:

- **name**
  Optional. This attribute specifies a name for the item group that can be uwsed to identify that group within a package.
- **subscription-element-id**
  Optional. This attribute specifies the persistent identifier of thegroup of elements within the subscription

6.2.2 Metadata

The `<icedel:metadata` element enables the entry of metadata on `<icedel:group`, `<icedel:add`, or `<icedel:remove-item` by using its attributes and description element.

Attributes on `<icedel:metadata` are shown in Figure 6.3.

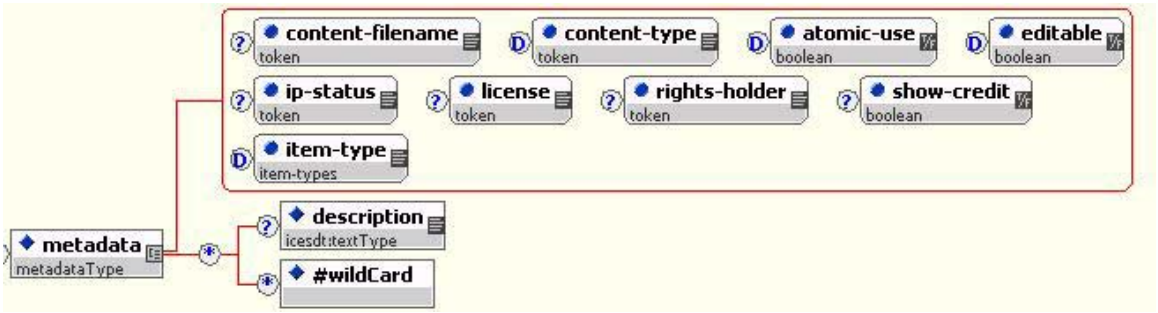

Figure 6.3 Attributes on Metadata element

- **content-filename**
  Optional. This element enables the specification the file name contained within.
- **content-type**
  Optional. This attribute enables the specification of the type of content such as "news."
- **atomic-use**
  Optional. This attribute specifies whether the element may be used in part. The values are Boolean, "true" or "false." The default is "false" because Basic ICE requires complete item usage.
- **editable**
  Optional. This attribute specifies whether the element is editable or whether it must be used as delivered. The values are Boolean, "true" or "false." The default is "false" because Basic ICE requires unaltered item usage.
- **ip-status**
  Optional. This specifies the intellectual property right status. The value is a token.
- **license**
  Optional. This specifies the license status of the content. The value is a token.
- **rights-holder**
  Optional. This attribute specifies the rights holder. The value is a token
- **show-credit**
  Optional. This attribute specifies the requirement to show credit for the content. The values are Boolean, "true" or "false." There is no default.
- **item-type**
  Optional. This attribute specifies a URI that identifies what type of item this is. For example the value of item-type may be "`http://icestandard.org/ICE/V20/item-type/offer`".

6.2.3 Add

The `<icedel:add` element is used to add new content according to the delivery policy of the subscription. It enables the attachment of metadata to the content being added. The

<icedel:add enables content to be directly included in the message by using the <icedel:item element, an indirect reference to content using <icedel:item-ref mechanism.
The structure of <icedel:add is shown in Figure 6.4.

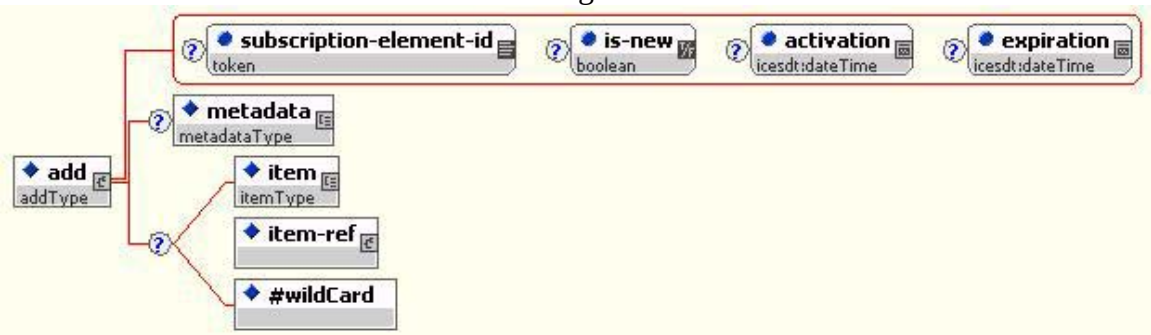

Figure 6.4 Add Element Structure

The <icedel:add element includes the following attributes:

- **subscription-element-id**
  Optional. This attribute specifies the persistent identifier of an element of a subscription that is being added. This applies to the contained item, item-ref or syndicator supplied (#wildcard) content.
- **is-new**
  Optional. This attribute specifies that the content is new. The values are Boolean, "true" or "false." There is no default.
- **activation**
  Optional. This attribute specifies when the addition for content is activated. The value is in the icesdt:dateTime format.
- **expiration**
  Optional. This attribute specifies when the content expires. This attribute specifies when the addition for content is activated. The value is in the icesdt:dateTime format.

6.2.4 Remove Item

The <icedel:remove-item element is used to remove content of the subscription.
The structure of <icedel:remove-item is shown in Figure 6.5.

Figure 6.5 Remove-item Structure

The <icedel:remove-item element has a single required attribute that identifies what is to be removed:

- **subscription-element-id**
  Required. This attribute specifies the persistent identifier of an element of a subscription

6.2.5 Item

The <icedel:item element directly carries content from the Syndicator to the Subscriber. An <icedel:item can carry the ICE <icesub:offer. The <icedel:item does not carry subscription management elements such as <icesub:subscribe or <icesub:cancel. The <icedel:item structure is shown in Figure 6.6.

Figure 6.6 Item Structure

The <icedel:item has two attributes:

- **content-transfer-encoding**
  Default. This attribute specifies the transfer encoding. Choices are base64 or x-native-xml with x-native-xml as the default.
- **name**
  Optional. This attribute specifies the item name that can be used as a transient identifier within a group or add.

6.2.6 Item-Ref

The <icedel:item-ref element references Syndicator content. The <icedel:item-ref structure is shown in Figure 6.7. It is made up of a single <icedel:reference element. This means that for each reference, an <item-ref element must be used.

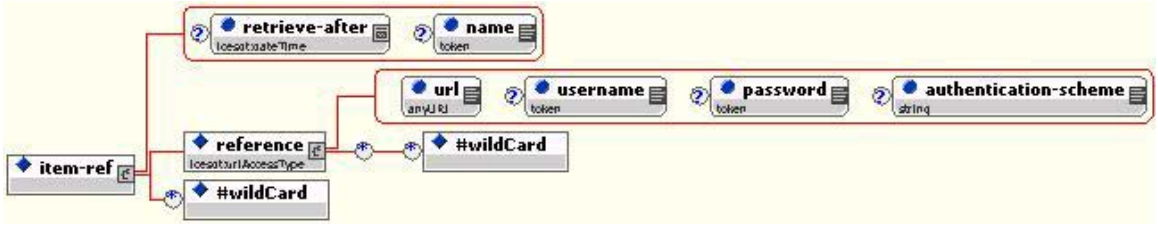

Figure 6.7 Item-ref Structure

The <item-ref element has two attributes:

- **retrieve-after**
  Optional. This attribute specifies a time after which the item can be retrieved. It is specified in the icesdt:dateTime format.

- **name**
  Optional. This attribute specifies the item name that can be used as a transient identifier within a group or add.

6.2.7 Reference

The `<icedel:reference` element is used to reference the content of the `<icedel:item-ref` element. The reference element is empty (with the exception of any wildcard content). The attributes carry the information for this element.
The `<icedel:reference` element has four attributes:

- **url**
  Required. This attribute specifies the URL from which the content can be retrieved.
- **username**
  Optional. This attribute specifies the username for retrieving the content if a login is required.
- **password**
  Optional. This attribute specifies the password for retrieving the content if a login is required.
- **authentificationscheme**
  Optional. This attribute specifies the authentification scheme for retrieving the content if this is required.

6.3 Package Confirmations

Full ICE has a mechanism to confirm the delivery of packages. If the package that is being delivered has `confirmation="yes"` then the Full ICE Subscriber must return a package confirmations response. The `<icedel:package-confirmations` element can contain one or more `<icedel:confirmation`. See Figure 6.8.

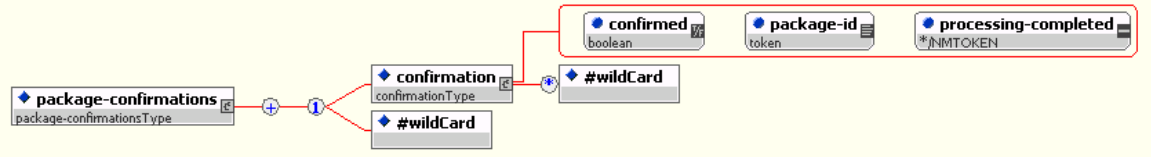

Figure 6.8 Package Confirmations Structure

Each `<icedel:confirmation` has the following attributes:

- **confirmed**
  Required. This attribute specifies whether the package delivery is confirmed. The value is Boolean and there is no default.
- **package-id**
  Required. This attribute specifies the unique id of the package within a subscription for which delivery is confirmed.
- **processing-completed**
  Optional. This attribute specifies whether the package was simply received or

whether it was processed. Values are "`received`" and "`processed`". There is no default.