Seminar „DB-Aspekte des E-Commerce“
SS 2001

Information and Content Exchange (ICE) Protokoll

Michael Kapp

1 Einleitung

Viele Internetanbieter stehen vor der Aufgabe, den Besuchern ihrer Seiten möglichst vielfältige und aktuelle Informationen liefern zu müssen. Oft werden spezielle Inhalte benötigt, die nicht selbst entwickelt werden können, sondern zugekauft und in die eigene Webseite integriert werden müssen. Geläufige Beispiele wären eine Finanzseite, die ihren Newsticker mit aktuellen Meldungen eines Börseninformationsdienstes versorgt, oder ein Online-Magazin, das neben eigenen Beiträgen immer den neuesten Wetterbericht bietet. Die Vermarktung von Inhalten an Geschäftspartner, im Englischen *content syndication* genannt, ist nicht neu: Nachrichtenagenturen oder Wetterdienste leben seit Jahrzehnten vom Verkauf ihrer Informationen [19].
Mit der Entwicklung des Internets hat die Content-Syndication an Bedeutung gewonnen, da hier potenziell jeder, der über eine ausreichend große strukturierte Sammlung digitaler Dokumente, Bilder, Videos oder sonstiger Informationen verfügt, als Datenanbieter (Syndicator) auftreten kann. Gerade im Bereich von E-Commerce und Business-to-Business (B2B) bieten sich viele Möglichkeiten. Um sich Wettbewerbsvorteile gegenüber Konkurrenten zu verschaffen, ist es für Online-Händler wichtig, dem Kunden detaillierte Informationen über die vertriebenen Produkte und eventuell noch Neuigkeiten aus der betreffenden Branche anbieten zu können. Eine der wichtigsten Informationsquellen sind die jeweiligen Hersteller, ob es nun um Datenblätter für bestimmte Bauteile oder die Ankündigung neuer Produkte geht.
Auf den ersten Blick scheint ein solcher Datenaustausch wenig komplex zu sein, in der Praxis bringt er jedoch zahlreiche neue Probleme mit sich. Die bisherigen Syndication-Lösungen sind oftmals Ad-hoc-Implementierungen und erfordern je nach Art der Geschäftspartner eine individuelle Programmierung. Das Zusammenspiel verschiedener Tools, wie FTP-Clients oder Perl-Skripte, und eventuell sogar manuelle Nachbearbeitung ist zur Filterung und Übermittlung der Inhalte nötig. Um den Überblick nicht zu verlieren und Daten strukturiert speichern zu können, setzen Firmen zunehmend auf Content-Management-Systeme, die mit den Syndication-Systemen zusammenarbeiten müssen. Die Empfänger beziehen nicht selten Material von verschiedenen Syndicatoren mit jeweils unterschiedlichen Übertragungsverfahren. Die Integration all dieser Verfahren ins eigene System bringt technische und wirtschaftliche Nachteile mit sich, bei den Datenabnehmern wie bei den Syndicatoren. Je mehr Geschäftspartner einbezogen sind, desto mehr werden die Mitarbeiter von Detailfragen und Problemen der Datenübertragung aufgehalten.
Ohne standardisierte strukturierte Datenformate und Protokolle ist es somit schwierig, den steigenden Informationsbedarf im Internet zu bewältigen.

Aus dieser Situation heraus entstand die Initiative von mittlerweile etwa 80 Firmen, das *Information and Content Exchange* (ICE) Protokoll zu entwickeln. Es wurde mit dem Ziel entworfen, einen einheitlichen Standard für die regelmäßige Lieferung von digitalen Inhalten über das Internet auf der Basis der Dokumenten-Beschreibungssprache XML zu schaffen und soll als breite Grundlage für unterschiedliche Formen der Content-Syndication dienen.

Für die Spezifikation des ICE-Protokolls [1] ist die sogenannte ICE Authoring Group verantwortlich. Die Gruppe setzt sich aus Mitarbeitern von Entwickler- und Anwenderfirmen von Content-Syndication zusammen. Führend bei der Entwicklung war Vignette, eine Firma aus dem Bereich E-Business-Software. Weitere Beteiligte sind unter anderem Sun, Adobe, Microsoft, CNET und National Semiconductor.

Die Version 1.0 der ICE-Spezifikation wurde im Oktober 1998 als eine der ersten XML-Anwendungen vorgestellt und zur Diskussion an das World Wide Web Consortium (W3C) [2] weitergeleitet. Die inzwischen vorliegende Version 1.1 enthält kleine Korrekturen und Erweiterungen.

2 Überblick

2.1 Einordnung des Protokolls

Das ICE-Protokoll regelt die Übertragung von Inhalten zwischen Datenservern, wobei es speziell auf Content-Syndication-Szenarios ausgerichtet ist. Das Protokoll weist den Kommunikationspartnern unterschiedliche Rollen zu: Eine ICE-Verbindung besteht immer zwischen einem Anbieter und einem Abnehmer von Informationen, in der ICE-Terminologie *Syndicator* und *Subscriber* genannt.
ICE basiert auf XML. Es verwendet XML jedoch nur dazu, um die Struktur der Protokollnachrichten zu definieren, nicht um die Struktur der übertragenen Inhalte festzulegen. Über das verwendete Datenformat macht ICE keine Aussagen. Die Interpretation, Speicherung und Weiterverarbeitung der empfangenen Daten sind Sache der Implementierung und laufen außerhalb des Protokolls ab. ICE beschränkt sich auf die Regelung der Kommunikation zwischen den Servern. Dadurch ist es besonders flexibel, was es als Grundlage für branchenspezifische Lösungen interessant macht.

Wichtig ist außerdem zu beachten, dass ICE nur die technischen Aspekte der Content-Syndication behandelt und keinen Ersatz für das Aushandeln einer Geschäftsbeziehung zwischen Syndicator und Subscriber darstellt. Finanzielle, juristische und ähnliche Fragen müssen anderweitig geklärt werden. Erst wenn die vertraglichen Dinge zwischen den Partnern ausgehandelt sind, kann die technische Umsetzung mit Hilfe von ICE beginnen.

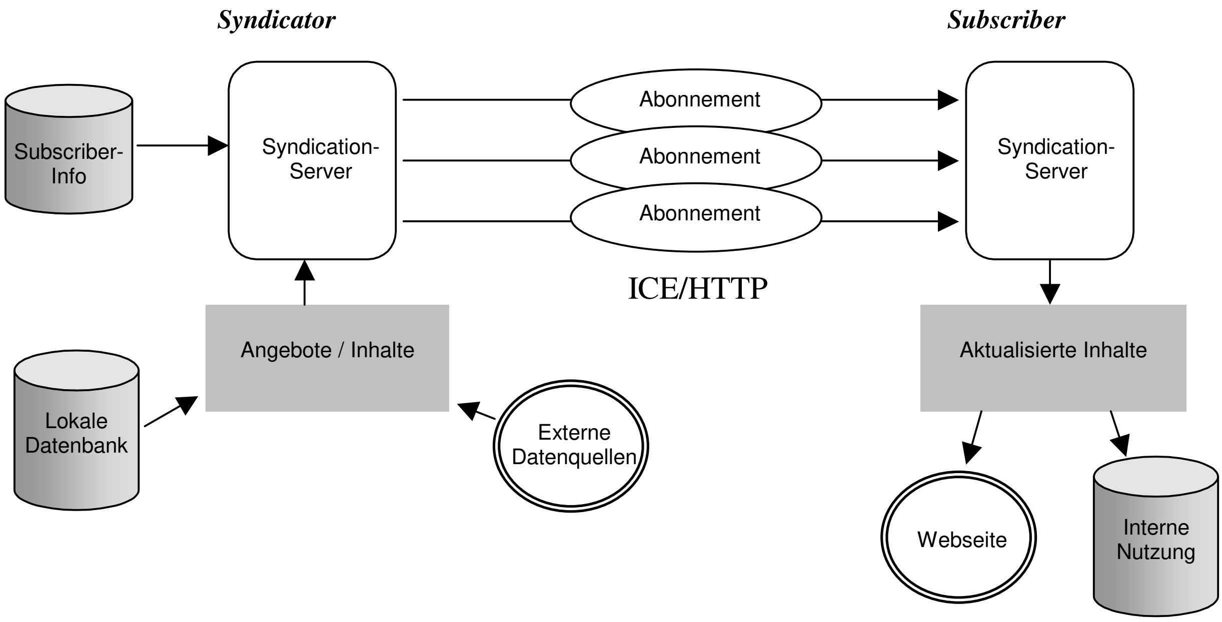


Abbildung: Typische Struktur einer Syndication-Lösung mit ICE

Als Grundlage von ICE wird ein verlässliches verbindungsorientiertes Transportprotokoll benötigt. ICE selbst ist ein Anwendungsprotokoll und prinzipiell unabhängig von der Transportebene. Die Spezifikation geht jedoch davon aus, dass im Normalfall HTTP zum Einsatz kommt, da dies auf nahezu jedem Webserver problemlos zu Verfügung steht. Bei der Verwendung von ICE über HTTP-Verbindungen [3] wird das weiter unten beschriebene Request/Response-Modell von ICE auf den HTTP-POST/Response-Mechanismus abgebildet.

2.2 ICE und andere Standards

Um Informationen von einer Website an eine andere weitergeben und -verwenden zu können, werden zwei Dinge benötigt:

1. ein gemeinsames Vokabular, um die Informationen zu verstehen
2. ein gemeinsames Kommunikationsprotokoll zur Übertragung der Daten

Viele Standards, die in der letzten Zeit entwickelt wurden, beschäftigen sich mit dem ersten Punkt, der Definition domänenspezifischer Vokabulare, meist mit Hilfe von XML. Beispiele wären PRISM (*Publishing Requirements for Industry Standard Metadata*), ein Standard zur Beschreibung von Metadaten für alle Arten von Publikationen, *XMLNews* für Nachrichten, WDDX (*Web Distributed Data Exchange*) für Datenstrukturen oder *RosettaNet* für Geschäftsprozesse [4].
ICE ergänzt diese Standards um den zweiten Punkt, ein Kommunikationsprotokoll für Syndication-Anwendungen. Es stellt also keine Konkurrenz dar, vielmehr können zum Beispiel nach PRISM formatierte Daten Inhalt eines ICE-Pakets sein.
Die folgende Darstellung zeigt diese Kapselung noch einmal am Beispiel der Übertragung von Online-News:

Transportprotokoll			
HTTP	**Anwendungsprotokoll**		
FTP	*ICE*	**Metadaten**	
		XMLNews-Meta	**Inhalt**
		PRISM	*XMLNews-Story*
			NITF
			JPEG

Auch verteilte Objektarchitekturen wie CORBA oder DCOM arbeiten teilweise auf einem anderen Level als ICE. Bei ihnen stehen Objektinterfaces und die Interaktion zwischen Objekten im Vordergrund. ICE könnte als mögliches Datenübertragungsprotokoll zwischen entfernten Objekten in solche Modelle integriert werden.
Mehr in Richtung der Funktionalität von ICE bewegt sich das *Channel Definition Format* (CDF) von Microsoft, das den Betrieb sogenannter *push channels* regelt. ICE baut auf einigen CDF-Konzepten wie etwa Lieferplänen auf, geht aber mit seiner Abonnement-Verwaltung und dem Paketsequenz-Modell weit über das hinaus, was CDF leistet. CDF wurde primär für die Verwendung durch den Endanwender konzipiert, während ICE auf den professionellen Austausch größerer Datenmengen ausgerichtet ist.
Ernstzunehmende Konkurrenz könnte ICE eventuell durch die Entwicklung von SOAP (*Simple Object Access Protocol*) drohen. Ursprünglich als Protokoll für *Remote Procedure Calls* (RPC) entworfen, eignet sich SOAP auch gut als allgemeines XML-basiertes Request/Response-Protokoll zwischen Servern. Um SOAP allerdings für Syndication wirklich interessant zu machen, werden noch zusätzliche, auf SOAP aufbauende Standards benötigt, welche die speziellen Aufgaben des Content-Handels regeln. ICE hat SOAP voraus, dass es

zahlreiche standardisierte Mechanismen bietet, die für erfolgreiche Syndication-Lösungen benötigt werden, wie zum Beispiel dedizierte Abonnements, individuelle Lieferregeln oder die Übermittlung von Nutzungsbedingungen.

2.3 Unterstützte Operationen

Die von ICE definierten Operationen lassen sich grob in drei Kategorien gliedern:

- Einrichtung und Verwaltung von Subscriptions
- Übertragung der Daten / Inhalte
- Administration (Ereignis-Logging, Benachrichtigungen)

Bei ICE erhalten die Subscriber die gewünschten Inhalte im Rahmen sogenannter *Subscriptions* (Abonnements) von den Syndicatoren. Aus der Sicht des Protokolls verläuft die Kommunikation zwischen einem Syndicator und einem Subscriber in zwei Stufen:

1) Zunächst muss eine Subscription eingerichtet werden. Hierbei werden die verschiedenen Parameter, wie etwa Zeitpunkt und Häufigkeit der Lieferungen, festgelegt. Eine nachträgliche Änderung ist möglich.

2) Danach beginnt die Phase der Datenlieferung. Abhängig von den vereinbarten Parametern werden die Datenübertragungen entweder vom Syndicator (*Push*-Verfahren) oder vom Subscriber (*Pull*-Verfahren) gestartet.

Zwischen einem Syndicator und einem Subscriber können beliebig viele Subscriptions bestehen. Außerdem verfügt ein Syndicator im Normalfall über mehrere Subscriber, ebenso wie ein Subscriber höchstwahrscheinlich Daten von verschiedenen Syndicatoren bezieht. Da alle Verbindungen zwischen Syndicatoren und Subscribern unabhängig voneinander sind, beschränken sich die weiteren Erläuterungen auf die Betrachtung der Kommunikation zwischen zwei Partnern.

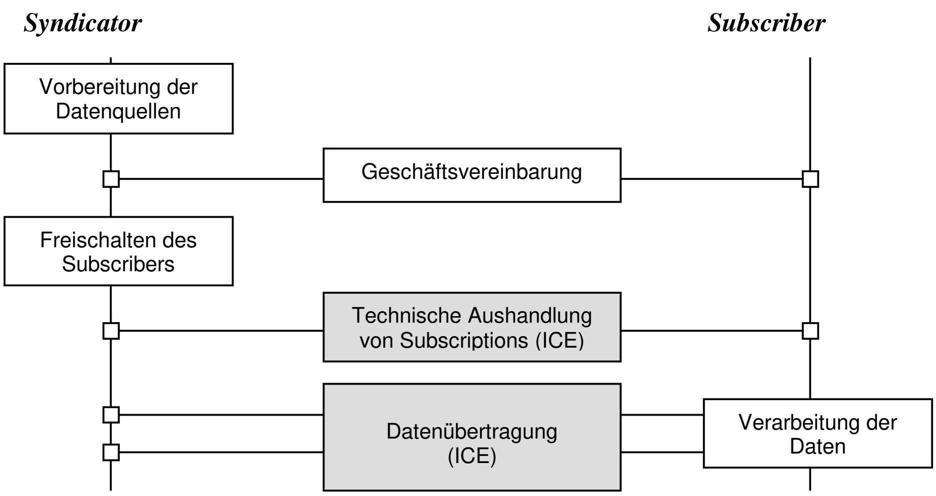


Abbildung: Verlauf einer Syndicator-Subscriber-Beziehung

2.4 Request/Response-Modell

Ein Hauptmerkmal von ICE ist, dass jede logische Operation durch ein Nachrichtenpaar aus einer Anfrage (*Request*) und einer entsprechenden Antwort (*Response*) repräsentiert wird [5]. Derjenige, der eine Operation initiiert, sendet einen Request an den betreffenden Partner. Dieser muss auf jeden Fall mit einer Response-Nachricht darauf reagieren, auch wenn für die jeweilige Operation logisch betrachtet eigentlich keine Antwort nötig ist. In solchen Fällen kann der Inhalt der Response leer bleiben. Eine gültige ICE-Protokollsitzung besteht demnach immer aus einer geraden Anzahl von Nachrichten.

Die typische Nachrichtenfolge von Anfrage – Antwort – Bestätigung besitzt bei ICE deshalb die Form: Request (Anfrage) – Response (Antwort) – Request (Bestätigung) – Response (leer).

Allen Requests und Responses müssen eindeutige IDs zugeordnet werden. Eine Response enthält außerdem immer einen Code, der über den Erfolg oder Misserfolg der Operation Auskunft gibt.

Eine einzige Nachricht kann mehrere verschiedene Requests enthalten, wobei die zugehörige Antwort dann eine Response zu jedem der Requests aufweisen muss. Eine Mischung von Requests und Responses innerhalb einer Nachricht ist nicht erlaubt, um die Komplexität zu verringern und einfache Implementierungen zu ermöglichen.

2.5 Payloads

ICE verwendet sogenannte *Payloads* als Grundlage des Datenaustauschs [5]. Eine ICE-Payload ist nichts weiter als ein gültiges XML-Dokument mit dem Root-Element „ice-payload“, das praktisch den Rahmen für alle ICE-Nachrichten bildet. Jede gesendete ICE-Nachricht (Request oder Response) entspricht genau einer Payload. Die ICE-Syntax ist über eine XML-DTD (Document Type Definition) definiert. XML Schema und Namespaces werden von der Spezifikation noch nicht berücksichtigt, was sich vermutlich in zukünftigen Versionen ändern wird.

Die Struktur einer Payload sieht typischerweise wie folgt aus:

```
<?xml version="1.0"?>
<!DOCTYPE ice-payload SYSTEM "http://www.mysite.org/ice.dtd">
<ice-payload
       payload-id="PL-2000-07-21T02:02:23.007-JoeCool-39"
       timestamp="02:02:23,449" ice.version="1.1" >
     <ice-header>
          <ice-sender
            sender-id="4af37b30-2c35-11d2-be4a-204c4f4f5020"
            name="Joe Cool" role="subscriber" />
          <ice-user-agent>IceBlock V7.0</ice-user-agent>
     </ice-header>

     ... (hier folgt die eigentliche Nachricht, Request oder Response)

</ice-payload>
```

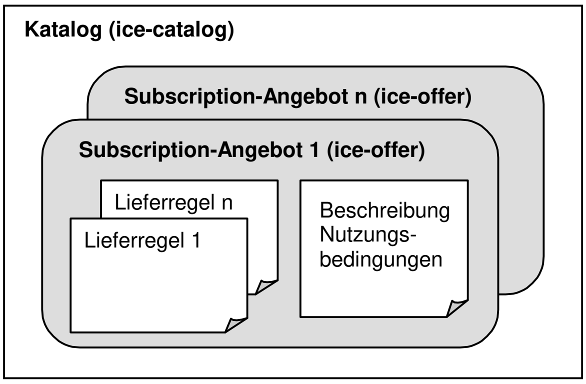
3 Einrichtung von Subscriptions

3.1 Anforderung des Katalogs

Bevor ein Subscriber mit Hilfe von ICE regelmäßig Inhalte von einem Syndicator empfangen kann, muss er zunächst eine Subscription (ein Abonnement) einrichten [6]. Voraussetzung dafür ist, dass der Subscriber einen Zugang zum ICE-Server des Syndicators erhalten hat, worauf ICE keinen Einfluss hat. Der Subscriber trägt die notwendigen Zugriffsdaten in sein ICE-System ein und muss zunächst einen Katalog mit den verfügbaren Subscriptions vom Syndicator anfordern. Dies geschieht über eine ICE-Payload, die einen `ice-get-catalog`-Request enthält. Zum Beispiel:

```
<ice-payload ... > ...
     <ice-request request-id="2001-05-27T14:11:09_TQXX7" >
          <ice-get-catalog />
     </ice-request>
</ice-payload>
```

Im ICE-Katalog listet der Syndicator alle Subscriptions auf, die der jeweilige Subscriber abonnieren darf, beispielsweise basierend auf den vertraglichen Vereinbarungen. Der Katalog soll kein Ersatz für eine Webseite sein, die über mögliche Abonnements informiert. Er dient einem ICE-Tool oder seinem Benutzer als Einstiegspunkt, um eine Subscription einzurichten.



Der Katalog (`ice-catalog`) wird vom Syndicator als Antwort auf den Katalog-Request gesendet und enthält für jede verfügbare Subscription ein Angebot (`ice-offer`) mit den Details. Die wichtigsten Bestandteile einer `ice-offer` sind die Beschreibung der Subscription, sowie die vom Syndicator beabsichtigten Lieferregeln (`ice-delivery-rules`).
Die Lieferregeln beschreiben, wann und wie häufig der Syndicator Updates für die jeweilige Subscription liefert und in welchem Modus (push oder pull) die Lieferungen erfolgen sollen. Durch die Kombination verschiedener Regeln in einem Angebot lassen sich sehr flexible Lieferpläne erstellen.

Nachfolgend das Angebot für einen Newsletter, der an Wochentagen zwischen 8 und 20 Uhr minimal 1 und maximal 10 mal im Push-Verfahren an den Subscriber übertragen werden soll. Am Wochenende soll die Abfrage im Pull-Verfahren stattfinden, wobei dann das Intervall für Updates mindestens 2 und höchstens 24 Stunden beträgt. Die Subscription ist gültig bis zum 31.12.2001.

```
<ice-payload ... >
    ...
    <ice-response response-id="RESP-20010527141123001" ... >
        ...
        <ice-catalog name="Newsmaker Newsletters" ... >
            ...
            <ice-offer
                    product-name="Newsmaker Inc. TechNews 3000"
                    description="Technology Newsletter"
                    offer-id="NEWS-189811-A"
                    subscription-id="ICE-NEW-SUBSCRIPTION"
                    expiration-date="2001-12-31" >
                <ice-delivery-policy
                        startdate="2001-05-20"
                        stopdate="2001-12-31" >
                    <ice-delivery-rule
                            mode="push"
                            weekday="1 2 3 4 5"            (Montag-Freitag)
                            starttime="08:00:00"
                            duration="P43200S"             (12 h)
                            min-num-updates="1"
                            max-num-updates="10"
                            />
                    <ice-delivery-rule
                            mode="pull"
                            weekday="6 7"                  (Samstag/Sonntag)
                            min-update-interval="P7200S"   (2 h)
                            max-update-interval="P86400S"  (24 h)
                            />
                </ice-delivery-policy>
            </ice-offer>
        </ice-catalog>
    </ice-response>
</ice-payload>
```

Zusätzlich zu den Lieferregeln lassen sich in einer `ice-offer` über verschiedene XML-Attribute noch weitere Hinweise auf die Nutzungsbedingungen setzen. Der Subscriber kann zum Beispiel darauf hingewiesen werden, ob er die gelieferten Inhalte verändern oder aufteilen darf, oder ob er sie nur unter Nennung der Quelle veröffentlichen darf. Zur Angabe weiterer Informationen, wie etwa Lizenzierungsbedingungen, können außerdem noch spezielle `ice-business-term`s zum Angebot hinzugefügt werden.
Die Bedingungen sollen ICE-Tools und Benutzern als Hinweis dienen, das Protokoll kann den Subscriber selbstverständlich nicht zur Einhaltung zwingen.

3.2 Auswahl und Verhandlung eines Angebots

Ist der Subscriber mit einem Angebot des Syndicators einverstanden, so kann er es annehmen, indem er dem Syndicator einen Request mit einer Kopie der `ice-offer` aus dem Katalog schickt. Der Syndicator überprüft die gesendeten Daten, und falls sich in der Zwischenzeit keine Änderungen ergeben haben und das Angebot noch gültig ist, antwortet er mit einer `ice-subscription`-Response, die das betreffende `ice-offer` nochmals wiederholt und dem Subscriber seine individuelle Subscription-ID mitteilt. Die Subscription ist damit eingerichtet.

In manchen Fällen sind die vom Syndicator im Angebot vorgeschlagenen Lieferregeln für den Subscriber nicht ohne weiteres akzeptabel. Um seine Netzlast besser zu verteilen, könnte der Subscriber etwa Lieferungen zu anderen Tageszeiten vorziehen.

Wenn der Syndicator bei manchen Parametern Verhandlungsspielraum hat, so sollte er dies durch `ice-negotiable`-Einträge innerhalb des Angebots anzeigen. Ein `ice-negotiable`-Element gibt die Menge von verhandelbaren Werten für einen speziellen Parameter an, wie im folgenden Beispiel eine Startzeit zwischen 4 und 7 Uhr:

```
<ice-negotiable type="ice-operation" id="starttime" >
     <ice-range id="starttime-builtin-definition" order="time" >
          <ice-span id="starttime" order="time" select="zero-or-one" >
               <ice-default-value> 04:00:00 </ice-default-value>
               <ice-span-min>      04:00:00 </ice-span-min>
               <ice-span-max>      07:00:00 </ice-span-max>
          </ice-span>
     </ice-range>
</ice-negotiable>
```

Falls der Subscriber verhandeln möchte [6], so kann er dem Syndicator einen Request mit einer nach seinen Vorstellungen modifizierten `ice-offer` senden. Die Modifikationen sollten im Rahmen der von den `ice-negotiable`-Elementen vorgegebenen Bereiche liegen. Da ICE dem Subscriber jedoch nicht vorschreibt, einen Katalog anzufordern, ist dies keine Pflicht.

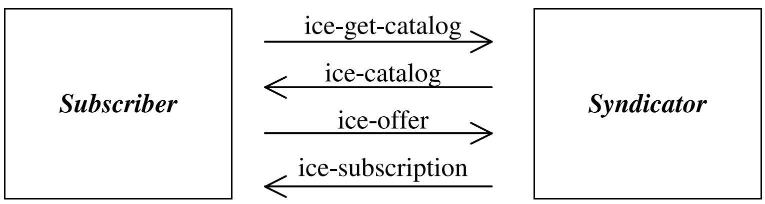
Der Syndicator kann das modifizierte Angebot

- *annehmen* und wie oben beschrieben mit einer `ice-subscription`-Response die Subscription einrichten.
- *ablehnen* und gleichzeitig ein neues *Gegenangebot* machen. Der Subscriber kann dieses Angebot aufgreifen, eine anderes aus dem Katalog auswählen oder die Verhandlung abbrechen.
- *ablehnen*, ohne weitere Hinweise. Der Subscriber kann dann einen neuen Katalog anfordern oder die Verhandlung abbrechen.

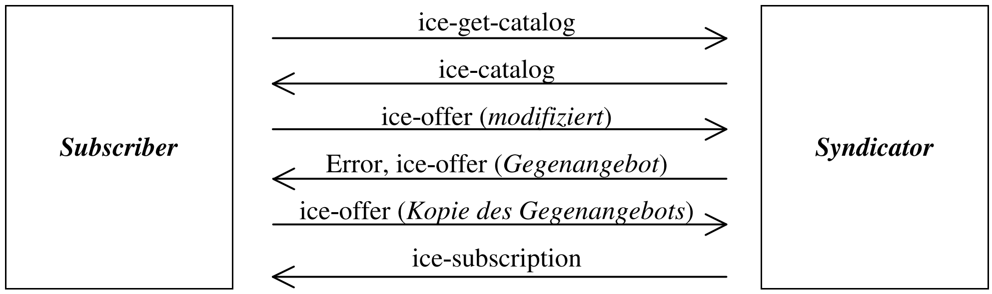
Die ICE-Version 1.1 sieht ein Verfahren zur automatischen Aushandlung der Parameter vor, indem gemeinsam akzeptable Werte durch sukzessive Gegenangebote eingekreist werden. Das Verfahren liefert entweder ein für beide annehmbares Ergebnis oder einen Fehler, falls keine Lösung existiert. Inwieweit dieses Verfahren genutzt wird, hängt von der Implementierung ab.

Der Subscriber ist nicht gezwungen, zu verhandeln oder Gegenangebote zu erstellen. In einer *Trivial Negotiation Implementation* kann er einfach sämtliche Angebote unverändert annehmen. Sollte der Syndicator aufgrund veränderter Bedingungen (etwa zwischenzeitliche andere Subscriptions) das zuvor unterbreitete Angebot plötzlich ablehnen, so kann ein neuer Katalog angefordert werden.

Abschließend noch zwei Beispiele zur Veranschaulichung des notwendigen Nachrichtenflusses. Die erste Darstellung zeigt den einfachsten Fall, die sofortige Annahme eines Angebots.



Als zweites Beispiel eine Verhandlung über zwei Runden (der Syndicator legt ein Gegenangebot vor und der Subscriber nimmt dieses an). Hier werden zwei Nachrichten mehr benötigt.



3.3 Statusoperationen

Zur Verwaltung bestehender Subscriptions (Identifikation jeweils anhand der ID) stellt ICE folgende Operationen [7] zur Verfügung:

- `ice-cancel` löscht eine bestehende Subscription. Kann sowohl vom Subscriber als auch vom Syndicator gesendet werden.

- `ice-change-subscription` ändert eine Subscription und führt zur Neuverhandlung der Parameter, indem `ice-offer`s wie oben beschrieben ausgetauscht werden. Auch diese Operation kann sowohl vom Subscriber als auch vom Syndicator initiiert werden. Das erste Angebot mit Angabe der verhandelbaren Parameter kommt allerdings immer vom Syndicator.

- `ice-get-status`: Hiermit kann der Subscriber den aktuellen Status einer Subscription beim Syndicator abfragen. Der Syndicator antwortet mit einer `ice-status`-Response, die unter anderem ein entsprechendes `ice-offer`-Element mit der Abonnement-Beschreibung enthält.

4 Datenlieferung

4.1 Pakete

Nachdem Subscriptions wie im vorigen Kapitel beschrieben eingerichtet wurden, konzentriert sich die Beziehung zwischen Syndicator und Subscriber hauptsächlich auf den eigentlichen Zweck von ICE, die regelmäßige Lieferung von Inhalten an den Subscriber (im Rahmen der zuvor ausgehandelten Lieferregeln).
Grundlage der Datenlieferung bei ICE sind Pakete. Eine Subscription besteht aus einer Folge von Paketen, die im Laufe der Zeit versendet werden. Ein Paket (`ice-package`) besteht aus den eigentlichen Inhalten in Form sogenannter *Items*, sowie aus Metadaten wie zum Beispiel dem erlaubten Veröffentlichungszeitpunkt.
Ein Item enthält Daten wie eine HTML-Datei oder eine Grafik. Die Daten können entweder innerhalb eines `ice-item` direkt in das Paket eingebunden werden, oder es wird mit Hilfe eines `ice-item-ref`-Elements eine Referenz (URL) zu dem gewünschten Objekt im Internet

übergeben [8]. Die Struktur der Daten ist beliebig, sie müssen jedoch in XML-konformer Codierung vorliegen. Binärdaten können *base64*-codiert werden. Die Referenzierung bietet sich bei Inhalten an, die sich schnell ändern. Die Zieladresse des `ice-item-ref`-Elements im folgenden Beispiel ist durch ein HTTP-Passwort geschützt, das dem Subscriber über einen zusätzlichen `ice-access`-Eintrag mitgeteilt wird. Daneben enthält das Paket noch ein `ice-item` mit direkt eingebundenen Daten. Auf die Bedeutung der Attribute „old-state" und „new-state" wird im nächsten Abschnitt eingegangen.

```
<ice-package package-id="PK-20010502-A"
      subscription-id="WQ379981-3A618267BF"
      old-state="S-2000-05-01" new-state="S-2000-05-02" >
    <ice-item-ref item-id="BN0502"
          url="http://www.newsmakerinc.com/news/latest.html" >
        <ice-access>
            <ice-access-control control-type="password"
              user="TheSubscriber" password="mysecret" />
        </ice-access>
    </ice-item-ref>
    <ice-item item-id="IN-28912" name="useless" >
        This is some inline content.
    </ice-item>
</ice-package>
```

4.2 Paketsequenz-Modell

Ein Paket kann sowohl beliebig viele Items hinzufügen (`ice-item`/`ice-item-ref`) als auch wieder löschen (`ice-item-remove`). Der aktuelle Gesamtinhalt einer Subscription, *Collection* genannt, ergibt sich aus der sukzessiven Anwendung der Hinzufügungs- und Lösch-Operationen aller Pakete. ICE benutzt Pakete als atomare Einheit, um eine Collection zu verändern.
Der Paketstrom einer Subscription muss von den Beteiligten als eine streng geordnete Sequenz von Paketen (*package sequence*) betrachtet werden [8]. Die Änderungen eines Pakets dürfen erst ausgeführt werden, nachdem alle Pakete mit logisch niedrigeren Ordnungszahlen verarbeitet wurden.

Beispiel: Gegeben ist folgende Paketsequenz:

Paket Nr.:	**1**	**2**	**3**
Fügt hinzu:	Item A, Item B	Item C	Item D, Item E
Löscht:		Item A	Item C

Zum Zeitpunkt des Pakets Nr. 3 besteht die Collection dann aus den Items B, D und E.

Um die Collection eines Subscribers auf den neuesten Stand zu bringen, könnte ihm der Syndicator immer sämtliche Pakete der Sequenz senden. Intelligenter sind *inkrementelle Updates*, bei denen der Subscriber nur die jeweils neuen Pakete erhält. Dazu muss sich der Subscriber seinen jeweils letzten Zustand merken, also das zuletzt empfangene Paket. Die Zustände werden bei ICE durch *Package Sequence Identifier* gekennzeichnet. Der Startzustand, in dem sich der Subscriber direkt nach Einrichtung der Subscription befindet, besitzt den vordefinierten Identifier „ICE-INITIAL". Die Identifier-Strings für alle weiteren Zustände können frei vom Syndicator gewählt werden, beispielsweise anhand der Ordnungszahl, des Zeitstempels oder des Zugriffsschlüssels auf eine interne Datenbank. Der Subscriber muss sich den Identifier seines jeweils aktuellen Zustands merken, darf aber von keinem bestimmten String-Aufbau ausgehen.

Jedes Paket wird vom Syndicator mit zwei Identifiern gekennzeichnet: mit dem Identifier des Ausgangszustands, in dem sich der Empfänger befinden muss, bevor er das Paket anwendet, sowie mit dem Identifier des Zustands, in den er nach Verarbeitung des Pakets gelangt. Auf diese Weise wird sichergestellt, dass immer alle Pakete in der richtigen Reihenfolge abgearbeitet werden.

Als Beispielszenario folgende Paketsequenz:

Paket Nr.	alter Zustand (old-state)	neuer Zustand (new-state)
1	ICE-INITIAL	S-2001-05-01
2	S-2001-05-01	S-2001-05-13
3	S-2001-05-13	S-2001-06-09

Der Syndicator muss jedem seiner Package Sequence Identifier den jeweiligen Zustand der Subscription zuordnen können. Meldet sich ein Subscriber mit dem Zustand „S-2001-05-13“, sendet ihm der Syndicator Paket Nr. 3. Der Subscriber überprüft nochmals, ob er sich im richtigen Zustand für das Paket befindet, verarbeitet es anschließend und nimmt dann den im Paket angegebenen neuen Zustand „S-2001-06-09“ ein.
Kommt ein Subscriber zu diesem laufenden Abonnement hinzu, befindet er sich zunächst im Zustand „ICE-INITIAL“. Um auf den aktuellen Stand zu kommen, benötigt er die Informationen aus allen drei Paketen. Anstatt die Pakete Nr. 1 bis 3 einzeln zu senden, kann der Syndicator die Änderungen in einem einzigen Paket zusammenzufassen und so gegebenenfalls Datenvolumen sparen. Ein solches Paket hätte dann die Einträge „ICE-INITIAL“ und „S-2001-06-09“ für den alten bzw. den neuen Zustand.
Wird für ein Paket kein bestimmter Zustand des Subscribers vorausgesetzt, kann es der Syndicator mit dem speziellen Identifier „ICE-ANY“ kennzeichnen. Um den kompletten Inhalt einer Subscription auszutauschen, kann der Syndicator alle Items einzeln per `ice-item-remove` löschen oder alternativ im Paket das Attribut *fullupdate* auf *true* setzen.

Falls kleinere Datenbestände des Subscribers aus irgendeinem Grund zerstört werden sollten, beispielsweise durch das versehentliche Löschen von Files, muss er die Collection nicht unbedingt komplett neu übertragen, sondern kann versuchen, eine Kopie der betroffenen Items über die *Individual Asset Repair*-Funktion von ICE (`ice-repair-item`) anzufordern [9]. Abhängig vom Syndicator kann diese Funktion auch fehlschlagen, sie stellt also keinen Ersatz für notwendige Backups des Subscribers dar.

4.3 Ablauf einer Übertragung

4.3.1 Pull-Modus

In jeder Lieferregel muss entweder der Pull- oder der Push-Modus zur Datenlieferung ausgewählt werden. Im Pull-Modus [10] liegt es am Subscriber, die Datenübertragung im vereinbarten Zeitrahmen zu initiieren. Er versendet dazu einen `ice-get-packet`-Request mit der betreffenden Subscription-ID und seinem aktuellen Zustand:

```
<ice-get-package subscription-id="WQ379981-3A618267BF"
  current-state="S-2001-06-01" />
```

Wenn ansonsten keine Probleme vorliegen (etwa eine unbekannte Subscription-ID), überprüft der Syndicator anhand des Subscriber-Zustands, welche Pakete noch ausstehen und reagiert mit einer der folgenden Antworten:

- *OK*: Eine Response mit einem oder mehreren `ice-package`-Elementen.
- *Zustand ist bereits aktuell*: Der Subscriber besitzt bereits den neuesten Subscription-Zustand, es müssen keine weiteren Daten übertragen werden.
- *Unbekannter Zustand*: Der übergebene Identifier ist dem Syndicator nicht bekannt. In diesem Fall sollte der Subscriber versuchen, den aktuellen Status über eine spezielle Abfrage (`ice-get-status`) zu erfahren oder eine Benachrichtigung an den Syndicator schicken.

4.3.2 Push-Modus

Der Push-Modus [10] gibt dem Syndicator die aktive Rolle. Sobald neuer Content vorliegt, baut er eine Verbindung zum Server des Subscribers auf und sendet die neuen Pakete per ICE-Request. Aus der Response des Subscribers lässt sich lediglich schließen, ob die Pakete auch tatsächlich angekommen sind, jedoch nicht, ob das Paket vollständig verarbeitet werden konnte. Um sicherzugehen, dass die Änderungen erfolgreich übernommen wurden, sollte der Syndicator in diesem Modus explizite Bestätigungen (siehe folgender Abschnitt) anfordern.

4.3.3 Bestätigungen

Für den Syndicator kann es nützlich sein zu erfahren, ob der Subscriber die empfangenen Pakete und die darin enthaltenen Referenzen vollständig verarbeiten konnte und in den erwarteten neuen Zustand gewechselt ist. ICE ermöglicht eine solches Feedback durch Setzen des *confirmation*-Flags in Paketen [11]. Der Subscriber ist dann verpflichtet, dem Syndicator nach dem Erhalt der betreffenden Pakete eine Bestätigung (ein `ice-code`-Request) mit einem entsprechenden Fehler- bzw. Erfolgscode zu senden. Bei positiver Antwort kann der Syndicator davon ausgehen, dass der Subscriber alle Änderungen erfolgreich durchführen konnte.

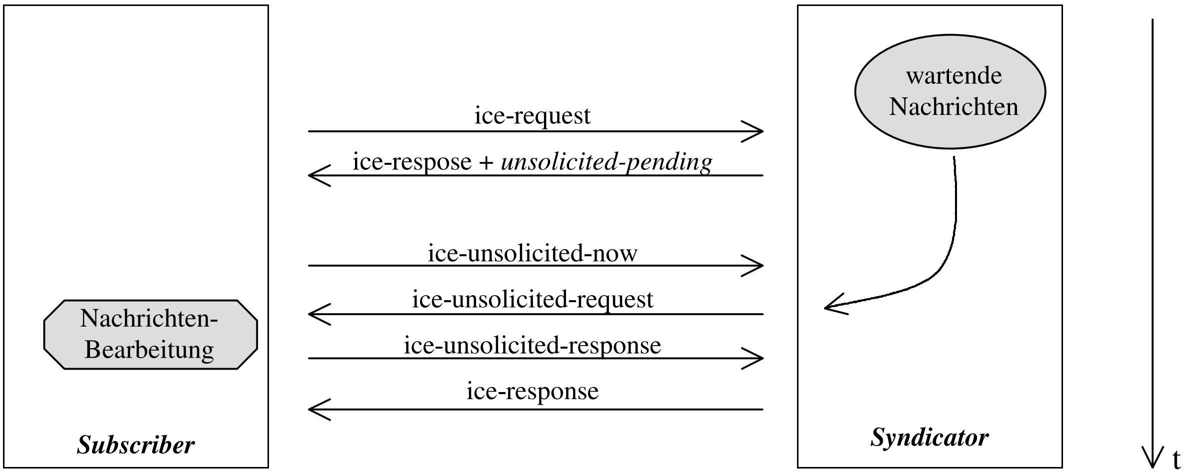
In einer Art einfacher Flusskontrolle kann der Syndicator mit dem weiteren Versenden von Paketen warten, bis er Bestätigungen für alle vorangehenden Pakete erhalten hat. Das Protokoll bietet außerdem die Möglichkeit, den Subscriber auf zu lange ausstehende Bestätigungen hinzuweisen (`ice-send-confirmations`).

5 Weitere Eigenschaften

5.1 Minimal Subscriber-Konzept

Während ICE davon ausgeht, dass ein Syndicator immer die notwendigen Ressourcen besitzt, um seinen Dienst rund um die Uhr zur Verfügung zu stellen, sind bei einem Subscriber auch Implementierungen ohne vorgesehen, die jeweils nur kurzzeitig aktiv oder nicht dauernd mit dem Netz verbunden sind. Ein solcher *Minimal Subscriber* besitzt keinen permanenten Server und kann nur Operationen initiieren, aber nicht auf plötzliche Requests des Syndicators reagieren. Die Datenlieferung an einen Minimal Subscriber kann deshalb nur im Pull-Modus durchgeführt werden.

Die Nachrichtenströme in ICE sind so ausgelegt, dass in der Regel der Subscriber als Initiator einer Verbindung auftritt. Dennoch gibt es ab und zu Fälle, in denen auch der Syndicator unaufgefordert eine Nachricht senden möchte, etwa um eine bevorstehende Systemwartung anzukündigen. Handelt es sich bei dem Gegenüber um einen „vollwertigen“, ständig erreichbaren Subscriber, läuft dies nach dem üblichen Request/Response-Schema ab.
Um auch einen Minimal Subscriber erreichen zu können, existiert in ICE das Konzept der *Unsolicited Messages* [12]:
Der Syndicator reiht alle Nachrichten für den Minimal Subscriber in eine Warteschlange ein. Sobald er wieder einmal einen (beliebigen) Request des Minimal Subscribers erhält, setzt er in der zugehörigen Response das Flag *unsolicited-pending*, das anstehende Nachrichten signalisiert. Der Subscriber fordert daraufhin mit einem `ice-unsolicited-now`-Request zur Übertragung auf. In der Antwort kann der Syndicator die Nachrichten nun als `ice-unsolicited-request`s senden. Die Bestätigung des Subscribers erfolgt über entsprechende `ice-unsolicited-response`s. Die folgende Abbildung veranschaulicht den Prozess noch einmal:



Das Besondere ist, dass die *unsolicited requests* aus der Sicht des Request/Response-Modells eigentlich eine Response darstellen, genauso wie die *unsolicited responses* einen Request. Um dem Request/Response-Modell zu genügen, muss der Syndicator deshalb die *unsolicited responses* nochmals mit einer normalen `ice-response` beantworten. Das gibt ihm aber nebenbei die Gelegenheit, das *unsolicited-pending*-Flag eventuell erneut zu setzen.
Der Subscriber entscheidet selbst, wann er auf ein gesetztes Flag reagiert. Der Syndicator kann jedoch die weitere Annahme von Requests verweigern, bis die Nachrichten abgearbeitet worden sind.

5.2 Austausch von Logs

Um Unregelmäßigkeiten verfolgen und Probleme besser diagnostizieren zu können, definiert ICE einen Mechanismus, über den *Ereignis-Logs* zwischen Subscriber und Syndicator ausgetauscht werden können [13]. Dieser Mechanismus besteht im wesentlichen aus einer Anfrage nach Logs (`ice-get-events`) und einer entsprechenden Antwort (`ice-events`), die soweit wie möglich mit Daten gefüllt werden kann. Die Spezifikation bietet ein eigenes Log-Format an, erlaubt aber auch die Benutzung von anderen Formaten.
Was im einzelnen geloggt wird, ist abhängig von der jeweiligen Implementierung und wird vom Protokoll nicht vorgeschrieben.

5.3 Protokollerweiterungen

Die ICE-Spezifikation erlaubt ausdrücklich implementierungsabhängige Erweiterungen des Protokolls [14] in den folgenden Bereichen:

- *Inhaltsmodell*: Erweiterung der Item- oder Paket-Definition in der XML-DTD, um XML-strukturierte Inhalte „nahtlos“ in Items bzw. Pakete einbetten zu können, wie im folgenden Beispiel das Element `article`:

```
<ice-item item-id="28">
    <article>
        <title>ICE, Syndication, and You</title>
        <author>I. C. Emerson</author>
        <abstract>This article discusses ICE.</abstract>
    </article>
</ice-item>
```

  Ohne eine Erweiterung der Item-Definition könnten zwar auch XML-Inhalte eingefügt werden, die von XML reservierten Steuerzeichen („<“, „>“, ...) müssten dann aber durch Escape-Sequenzen ersetzt werden, damit das Dokument weiterhin gültig bleibt. Mit der Ergänzung der DTD können die Inhalte direkt durch den XML-Parser extrahiert werden und müssen nicht zuvor noch konvertiert werden.

- *Parameterverhandlung*: Mit Hilfe von `ice-negotiable` lassen sich nicht nur Lieferregeln, sondern auch andere, nicht durch ICE definierte Parameter aushandeln. Ein solcher Parameter könnte bei einer Syndication von Videostreams die Qualität der gesendeten Daten sein, die je nach verfügbarer Bandbreite zu einem Subscriber festgelegt werden soll.

- *Spezifische Log-Formate*

- *Benutzerdefinierte Request-/Response-Nachrichten*: können bei der Einrichtung einer Subscription vereinbart werden

6 Vorteile und Grenzen

Die Vorteile des ICE-Protokolls liegen darin, dass es auf die Bedürfnisse des anvisierten Einsatzgebiets zugeschnitten ist und dennoch relativ offen und flexibel bleibt.

- Die zugrundeliegende Beschreibungssprache XML gilt als zukünftiger Standard für den strukturierten Datenaustausch und findet in immer mehr Bereichen Anwendung. ICE-Systeme können auf vorhandene XML-Tools zurückgreifen, wie zum Beispiel Parser, was die Handhabung des Protokolls erleichtert. Außerdem profitiert ICE von allen Vorteilen von XML, wie Lesbarkeit (Debugging), Erweiterbarkeit oder Modularität.

- ICE lässt sich problemlos in Verbindung mit bestehenden Transportprotokollen nutzen, speziell mit HTTP, das von nahezu allen Webservern unterstützt wird.

- Das Protokolldesign zielt auf einen minimalen Overhead, bezogen auf die Komplexität des Nachrichtenaustauchs und die notwendige Zustandsverwaltung.

- Durch die erweiterbare Architektur und die Abstraktion vom Datenformat ist es offen für alle Anwendungsgebiete und zukünftige, bisher nicht bekannte Entwicklungen. Über eine Ergänzung der Protokolldefinition lassen sich beliebige branchenspezifische XML-Datenformate einbinden.

- ICE geht explizit auf verschiedene Fragen der Content-Syndication ein, zum Beispiel die Aushandlung detaillierter Lieferpläne oder die Übertragung von Metadaten wie Nutzungsrechten oder Veröffentlichungszeitpunkten.

- Das Paketsequenz-Modell erlaubt flexible Datenlieferungen und garantiert die vollständige Übertragung aller Änderungen.

- Die weitgehende Automatisierung des Datenaustauschs ermöglicht gut skalierbare Lösungen.

Neben den zahlreichen Vorteilen bringt aber gerade die Offenheit von ICE auch einige Probleme bei der Anwendung mit sich:

- Das Protokoll enthält bislang keine Sicherheitsfeatures. Die Verschlüsselung sensitiver Daten muss auf anderen Ebenen geschehen, zum Beispiel mit Hilfe von HTTP/SSL oder PGP. Auch die Zugriffskontrolle durch passwortgeschützte Zugänge oder die Authentifizierung der Kommunikationspartner über Zertifikate geschieht typischerweise auf Transportebene.
  Die ICE Authoring Group verweist darauf, dass sich bereits verschiedene allgemeine Sicherheitsstandards, etwa zur Signierung von XML-Dokumenten, in Entwicklung befinden und man kein eigenes, eventuell inkompatibles System einführen wollte.

- Die finanzielle Seite einer Subscription, sprich die Bezahlung der gelieferten Informationen, bleibt bei ICE außen vor. Wegen der fehlenden Sicherheitsstandards kann das Protokoll kein elektronisches Payment-System ersetzen, für „pay-per-item“-Modelle wäre aber zumindest eine standardisierte Möglichkeit sinnvoll, einzelnen Datenpaketen oder Abonnements Preise zuordnen zu können.

- Die Unabhängigkeit vom Datenformat macht ICE universell, erfordert aber eine Abstimmung zwischen Subscriber und Syndicator und eventuell zusätzliche Konvertierungen.

- Die Verwendung von XML produziert zwangsläufig ein höheres Datenvolumen als bei gleichartigen binären Protokollen. Abhilfe stellt die Kompression der Datenpakete auf Transportebene dar. Für einfache Systeme mit begrenzten Ressourcen ist ICE aber eher ungeeignet.

7 Ausblick

Einige potenzielle Erweiterungen in zukünftigen Versionen des ICE-Standards werden in der aktuellen Spezifikation bereits angedeutet: Durch neuere Konzepte wie XML Schema und Namespaces wird sich die Integration benutzerdefinierter Datenformate in ICE besser formalisieren lassen, so dass sich hier Änderungen an der Spezifikation anbieten.
In der jetzigen Version unterstützt ICE bereits rudimentär die Aufstellung von Nebenbedingungen (*constraints*) für Subscriptions, ohne sich jedoch auf Sprachen festzulegen, die diese Nebenbedingungen beschreiben könnten. Wünschenswert wäre nach Ansicht der ICE Authoring Group eine Sprache, mit der sich Anforderungen wie zum Beispiel „in der Subscription enthaltene Anzeigen-Banner müssen im GIF-Format vorliegen und dürfen nicht größer als X × Y Pixel sein“ formal ausdrücken lassen. Da dazu momentan keine bekannte Sprache in der Lage ist, wird an den Entwurf einer eigenen Constraint-Sprache (*ICE Constraints*) gedacht [15].

In der Zwischenzeit bieten bereits zahlreiche Unternehmen E-Commerce-Lösungen mit ICE-Unterstützung an. Dazu gehören der *Vignette Syndication Server*, *Intershop Enfinity*, *Kinecta Interact*, *ArcadiaOne eSyndication*, *Microsoft BizTalk*, das *ShiftKey Syndication System* oder *HP Bluestone* [16]. Auch eine Open Source Java-Implementierung des Protokolls (*ICE Cubes*) ist in Entwicklung [17]. Der tatsächliche Einsatz von ICE scheint aber bislang trotzdem eher schleppend zu verlaufen, wie man Pressemeldungen entnehmen kann [18]. Ein Grund könnte darin liegen, dass sich viele Firmen über die neuen Möglichkeiten im Bereich Content-Syndication erst bewusst werden müssen.
Durch den Aufbau auf bestehenden Standards und die gebotenen Automatisierungsfähigkeiten besitzt ICE auf jeden Fall das Potenzial, einen Teil der nötigen Kommunikation im E-Business effizienter und kostengünstiger zu gestalten. Die Zukunft wird zeigen, ob es sich gegen die bereits etablierten Syndication-Verfahren und alternative, zum Beispiel auf SOAP basierende Lösungen langfristig durchsetzen kann.

Quellenangaben

[1] ICE-Spezifikation, Version 1.1: *www.icestandard.org/spec/SPEC-ICE1.01-20000511.html*
[2] Word Wide Web Consortium: *www.w3.org*
[3] ICE/HTTP: ICE-Spezifikation 1.1, Kap. 2.3
[4] vgl. *www.prismstandard.org*, *www.xmlnews.org*, *www.openwddx.org, www.rosettanet.org*
[5] Payloads, Request/Response: ICE-Spezifikation 1.1, Kap. 2.2.1-2.2.2
[6] Einrichtung und Verhandlung von Subscriptions: ICE-Spezifikation 1.1, Kap. 4.1-4.5
[7] Subscription-Statusoperationen: ICE-Spezifikation 1.1, Kap. 4.6
[8] Paketinhalt/Sequenz-Modell: ICE-Spezifikation 1.1, Kap. 5.1-5.2
[9] Individual Asset Repair: ICE-Spezifikation 1.1, Kap. 5.5.3
[10] Pull-/Push-Modus: ICE-Spezifikation 1.1, Kap. 5.3-5.4
[11] Bestätigungen: ICE-Spezifikation 1.1, Kap. 5.5.1
[12] Unsolicited Messages: ICE-Spezifikation 1.1, Kap. 3.10
[13] Ereignis-Logs: ICE-Spezifikation 1.1, Kap. 6
[14] Protokollerweiterungen: ICE-Spezifikation 1.1, Kap. 8
[15] ICE Constraints: ICE-Spezifikation 1.1, Kap. 1.4.1
[16] vgl. *www.icestandard.net/authoring_prodsrv.asp*
[17] Java ICE Cubes: *sourceforge.net/projects/icecubes*
[18] Chuck Moozakis: „ICE Keeps Data Fresh“; *www.internetweek.com/ebizapps01/ebiz050701-2.htm*
[19] Achim Wagenknecht: „Content-Syndication: Fakten im Abo“; *www.zdnet.de/internet/artikel/wdm/200103/inhalte_00-wc.html*