R&D White Paper

WHP 134

May 2006

Portable Content Format:

a standard for describing an interactive digital television service

R. Bradbury, R. Cartwright, J. Hunter, S. Perrott, J.E. Rosser

Research & Development
BRITISH BROADCASTING CORPORATION

BBC Research & Development
White Paper WHP 134

Portable Content Format – a standard for describing an interactive digital television service

R. Bradbury, R. Cartwright, J. Hunter, S. Perrott, J.E. Rosser

Abstract

The DVB's Portable Content Format (PCF) is a data format for the description of interactive digital television (iTV) services. It is intended to support the business-to-business interchange of interactive content and to enable deployment on multiple target platforms with a minimum amount of re-authoring. This will reduce both the production cost and time to air of iTV services.

White Papers are distributed freely on request.

Authorisation of the Chief Scientist is required for publication.

BBC Research & Development
White Paper WHP 134

Portable Content Format – a standard for describing an interactive digital television service

R. Bradbury, R. Cartwright, J. Hunter, S. Perrott, J.E. Rosser

1 Introduction

This paper is intended to provide an introduction to the DVB Portable Content Format (PCF)[1]. In addition to a general overview (section 2), it explains the significance of the PCF to the BBC (section 3) and details key characteristics of the format (section 4).

The BBC has been a major contributor to the development of the PCF specification, which was approved by the DVB Technical module in January 2006. Continuing research at BBC R&D convinces us of its credibility as a format for both describing an interactive television (iTV) service and enabling automatic translation into platform specific versions.

2 Overview

Today's digital television platforms offer a wide variety of interactive television services. To the casual observer, this may appear the result of mature production processes across the industry. However, the reality is that the majority of interactive content is developed in non-portable ways, specifically targeting individual platforms. The resulting high production costs mean that interactive content is often limited to high-profile programming and revenue generating propositions.

Using the PCF enables a host of new opportunities for the creation and delivery of interactive services by:

- increasing service portability and so reducing production costs;
- simplifying interchange of interoperable interactive content, both between content providers, and between content providers and platform operators;
- enabling content providers and platform operators to choose the best possible tool for their business at all points of the iTV production chain;
- enabling automatic translation into platform-specific formats leading to a significant reduction in cross-platform costs, stimulating new business models.

The PCF captures the intended viewer experience of the service author. This means that iTV services can be authored and deployed without the author needing to know about the viewer's interactive technology.

At the heart of the PCF is a data model that is independent of the format's representation. However, XML was identified as a suitable format for representing PCF services and a set of schemata for validating the PCF XML accompanies the PCF specification.

The PCF is dependent on a transformation step often referred to as "transcoding". This step converts a PCF service description into a platform-specific version. A PCF "transcoder" is free to make use of any of the features of a target platform, in any combination, to implement the authorial intent captured by a particular PCF feature.

A prototype transcoder has been developed at BBC R&D to prove that transcoding from PCF into a platform specific version is a viable procedure. The R&D transcoder takes PCF XML documents as input, validates them against the PCF schema, creates an internal data structure and then translates this into a platform specific output. The transcoder has already been successfully used to generate output for an MHEG application.

It is important to note that the PCF is not intended to be a receiver technology (or "middleware"); there is no requirement that the transcoding take place in the receiver. Neither is the PCF an authoring tool, although the output from an authoring tool could be PCF.

---

[1] This paper draws upon information from a number of other published documents [1], [2], [3] and [4].

3 Motivation

As a world leader in the deployment of iTV, the BBC began delivering interactive information services and enhanced television programmes in the UK back in 1998, and the number of services the BBC delivers has increased significantly year-on-year. However, producing iTV services is a non-trivial task due to the diversity of deployed digital television platforms and the lack of authoring tools that completely meet our needs.

3.1 Too many deployed DTV platforms…

The BBC is committed to providing iTV services to all its licence fee payers irrespective of the DTV platform they have chosen. But the different UK DTV platforms have deployed different interactive technology (often referred to as interactive middleware or API). This means that to provide the same interactive content for each platform, the BBC must author a version specific to each target platform.

This problem is not unique to the BBC or even the UK. A wide range of interactive technologies has been deployed in platforms around the world. This is because platform operators did not consider the portability of interactive content to be a primary concern when selecting their interactive technology. Instead, their choice was based on cost of hardware, time involved to deploy or upgrade, licensing terms and the complexity of integration with the operator's choice of Electronic Programme Guide (EPG) and (where required) Conditional Access (CA) systems.

This multitude of target platforms makes distribution of interactive services beyond primary markets impractical, or at the very least un-economical, as a substantial amount of re-authoring is normally required.

If only a single interactive technology were deployed in all platforms! The DVB Multimedia Home Platform (MHP) has been developed to be such a technology. However, given that the majority of digital television receivers deployed today neither support MHP nor have a hardware footprint that allows them to be upgraded to support MHP, this is not a likely outcome in the near future[2].

Even if the same underlying interactive technology did exist in all platforms, there are a number of other practical issues that complicate the task of authoring interactive content for delivery to multiple platforms, including:

- Physical constraints, such as a platform not having a return path;
- Commercial constraints, such as a provider lacking permission to access certain platform resources;
- Operational constraints, such as part of the screen being reserved for special platform features or branding;
- Inherent differences between platforms, such as the available fonts and text rendering rules of a platform.

Although the PCF cannot provide solutions for such intrinsic practical issues, it does preserve the viewer experience of a service very closely across platforms. In this way, it is analogous to the way in which a PDF reader will render Adobe's Portable Document Format (PDF) within the constraints of a particular system (e.g. the rendering of content on a Mac OS desktop computer, versus Windows CE on a PDA).

3.2 Too many iTV authoring tools…

There is already a vibrant market in authoring tools for interactive television (iTV) and the BBC has spent some time looking at various products from a range of vendors. Whilst many of these tools are mature, well-featured products, the BBC has not been able to exploit them as much as it would like. Some tools only generate output specific to a particular platform. Others need a specific application, effectively a specialised browser, to be broadcast and loaded onto a receiver before the output from the tool can be presented – requiring additional bandwidth for delivery. No single authoring tool appears capable of supporting all possible types of interactive service and some even struggle to provide for all aspects of a single service.

The result is that the BBC needs more than one authoring tool to produce its iTV services, and this increases the commitment of effort, time and investment in infrastructure. The complications of integrating multiple authoring tools into the production chain is magnified by the need to target multiple DTV platforms, as illustrated by the number of interfaces in Figure 1. The consequence is that the BBC, like some other broadcasters, is not always able to use the best tool for a particular job.

---

[2] Another possible solution would be to try and translate directly from one platform-specific coded version of an interactive service to another. However, research at BBC R&D has shown that this is not a practical option because at this low-level, the underlying data and programmatic models of interactive technologies are significantly different.

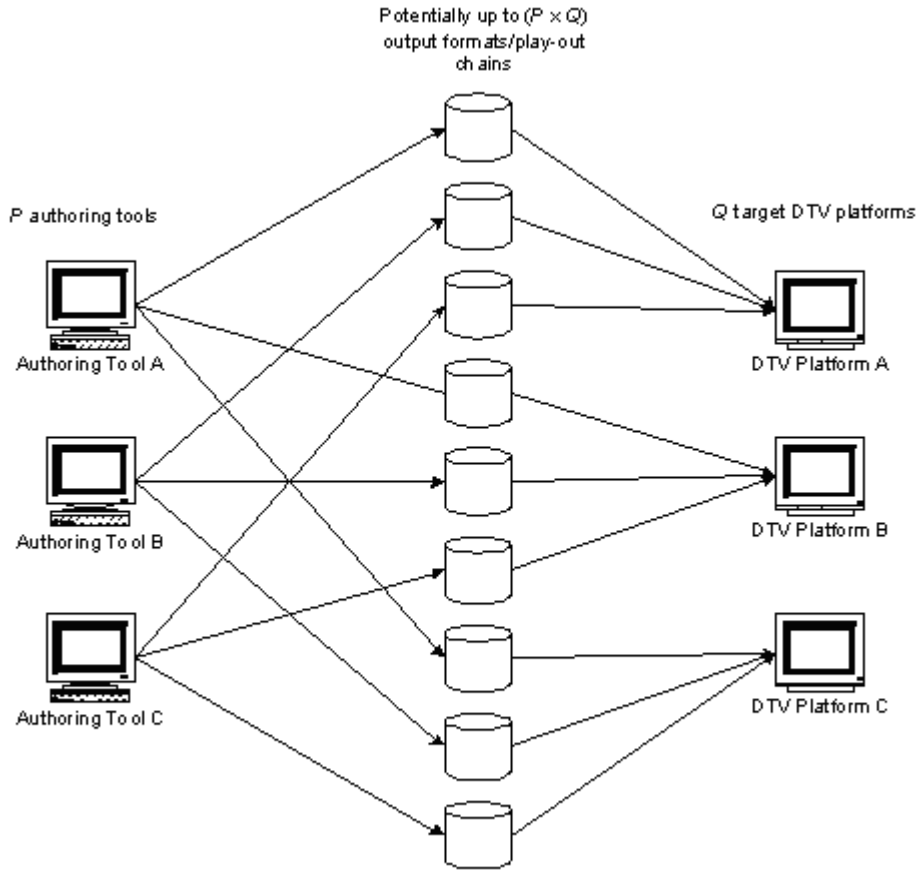

Figure 1 - Interfaces between multiple iTV authoring tools and platforms.

The PCF could be generated as an output from authoring tools. This would:

- reduce the integration effort required in adopting a new tool, so allowing a more unrestricted choice to be made from the marketplace;
- increase the likelihood of the best tool for a particular job being used, resulting in interactive content that is produced either more cheaply, more quickly or at a higher quality.

Figure 2 illustrates how PCF can be used to integrate a variety of authoring tools into a production chain in order to generate interactive services for a variety of platforms.

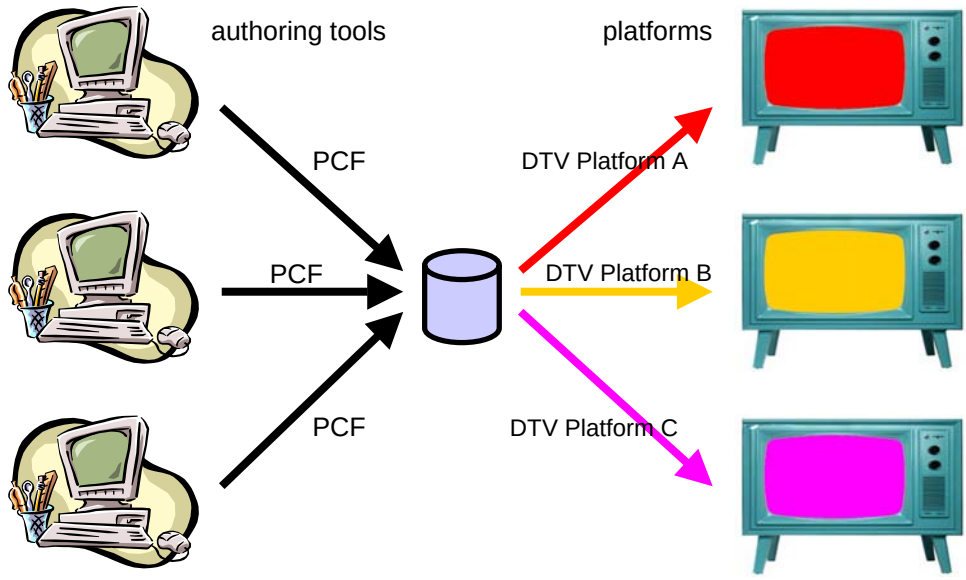

Figure 2 – PCF enables multiple authoring tools to produce output for multiple target platforms.

4 A service author's guide to PCF

The key characteristic of the PCF is that it enables the description of "what" the viewer experience should be, rather than "how" it should be achieved. This characteristic is essential if the inherent differences between different digital television platforms are to be successfully accommodated. The PCF accomplishes this by allowing the iTV service to be described using a high-level declarative syntax which enables the description of the intended viewer experience without prescribing exactly how it should be rendered on a target platform. This means that any automatic conversion process has sufficient freedom to transform the description into the execution model required by the target platform.

The structures within the PCF for the representation of data have been designed with consideration of many of the practical issues surrounding the production of interactive services. The PCF supports independent description of different aspects of the interactive service, i.e. **content**, **presentation** (layout and style), **behaviour** and **navigation**. Furthermore, the PCF does not require all aspects of a service to be described as one physical unit, such as a file. For example, service descriptions can be arbitrarily distributed across files located on the Internet, using references represented as Uniform Resource Identifiers (URIs).

These structuring techniques allow the most appropriate party, whether a person or an organisation, to take their role in the creation and management of a part of an interactive service. They also provide the means to efficiently manage any changes in the description of an interactive service, e.g. a real-time content update.

4.1 Range of services supported by the PCF

The PCF supports a wide range of service types, including those illustrated here.

Figure 3 shows an information-driven interactive service. Information is spread across a potentially large hierarchy of pages over which the viewer can navigate at their own pace. The information being presented may change over time and the service will update to reflect this.

Figure 4 shows an enhanced TV service that allows a viewer to play a simple game. The viewer is asked to categorise a number of items using the coloured keys on their remote control. After each attempt, an up-to-date score is displayed and the visual presentation of the game changes to reflect items correctly sorted. Sounds play in response to certain game events.

Figure 3 – Information

Figure 4 – Simple game

Figure 5 shows an interactive service where a viewer can donate money and register a vote, whilst continuing to watch video in a quarter screen area.

Figure 6 shows an enhanced TV service that allows a viewer to play along with a linear quiz or test. Questions are synchronised to a linear programme and have to be answered by the viewer pressing a colour key within a given timeframe. At the end of the show, the viewer's personal score is calculated and shown on the screen.

Figure 5 – Donating and voting

Figure 6 – Synchronized quiz

The PCF will be capable of representing many other kinds of service, including multi-stream sport services and information society services. However, the initial phase of specification has not considering support for:

- Reactive (or “twitch”) games;
- Text and web documents that have no iTV specific formatting;
- Target platforms that are not primarily digital television platforms, such as mobile phones and PDAs.

Those services that PCF cannot describe fully can still benefit in some way from the interoperable framework that the PCF will provide.

4.2 An overview of a PCF service description

The PCF is based on industry standard formats, including XML syntax, MIME types and the Unified Modelling Language (UML).

The following XML is an example of a very simple PCF service:

```
<PCF xmlns="http://www.dvb.org/pcf/pcf">
    <Service name="hello_world">
        <String name="pcfSpecVersion" value="1.0"/>
        <URI name="firstScene" value="#hello_scene"/>
        <Scene name="hello_scene">
            <Component class="TextBox" name="hello_text">
                <Size name="size" value="100 40"/>
                <String name="content" value="Hello, World!"/>
            </Component>
            <OnEvent name="exit">
                <Trigger eventtype="KeyEvent">
                    <UserKey name="key" value="VK_PREV"/>
                </Trigger>
                <ExitAction/>
            </OnEvent>
        </Scene>
    </Service>
</PCF>
```

Whilst this service is very small, it uses many of the key PCF structures. The **Service** element provides the entry point into the service, and the rest of the service description is either contained, or referenced from, within this element.

Immediately within the Service element is a single **Scene** element. All PCF services are sub-divided into one or more Scenes, and each Scene represents a destination that may be navigated to within the service. The visual appearance of a Scene is defined by the set of **Components** the Scene contains. In this case there is only a single component, a **TextBox**, which is used to display the message "Hello, World!". PCF defines a toolkit of standard components that may be included within a Scene to describe the desired appearance and behaviour.

All component types have a number of **Properties**. For the TextBox component, only its **size**, and its **content** are being explicitly set. A TextBox has a number of other properties that have not been explicitly set, and in this case their default values will be used.

The last element within the Scene describes an event handler, **OnEvent**. For this Scene the event handler responds to the viewer pressing the virtual key "VK_PREV" that will be mapped to a platform specific remote control key, e.g. the

"Backup", "Back", "Previous", or "Cancel" key. The event handler contains a single action, ExitAction, that instructs the service to exit.

Whilst this is a very simple service it does illustrate core aspects of a PCF service description:

- **Components** – the building blocks of a PCF service description (see section 4.3).
- **Content** – managed and presented using PCF components (see section 4.4).
- **Behaviour** – the response to events generated at run-time (see section 4.5).

4.3 Components

Components are the building blocks of a PCF service description. Many components are visual "widgets" used to describe the visual appearance of the service at a particular point, for example TextBox, Video and Menu. There are also non-visual components, which can be used for the presentation of non-visual content or providing other aspects of service functionality, for example Audio, Stream and Random.

All component types may be declared using a generic *component item*. This has a *class* property to specify the type of component

Each class of component has a set of associated named properties, and setting these properties defines how a particular instance of a component will look and behave. The following is an example of a standard PCF TextBox component declaration:

```
<Component class="TextBox" name="MainText">
    <Size name="size" value="100 40"/>
    <Color name="fillcolor" value="#0000AA"/>
    <Color name="textcolor" value="#FFFFFF"/>
    <String name="content" value="Hello, World!"/>
</Component>
```

In this case, four of the TextBox's properties are being explicitly set: its *size*, its *fillcolor* (background colour), its *textcolor* and the *content* string. As well as being defined during instantiation, many component properties can be dynamically changed at run-time. In the above example, the TextBox's *content* property can be updated to display a different block of text, or one of its *colour* properties can be updated to change that aspect of the component's presentation.

Using the generic *Component* provides a number of advantages regarding the extensibility of the PCF format, and the ability to declare custom components using the same syntax as standard PCF components. However, a number of standard components exist that may also be declared in a class specific way. For example, the following example describes a TextBox that is equivalent to the Component of the TextBox class:

```
<TextBox name="MainText">
    <Size name="size" value="100 40"/>
    <Color name="fillcolor" value="#0000AA"/>
    <Color name="textcolor" value="#FFFFFF"/>
    <String name="content" value="Hello, World!"/>
</TextBox>
```

Some Component types have intrinsic behaviour that will cause them to change in response to external events, such as key presses, broadcast stream events, or a change in state of the return path connection. For example, a Menu component will react to specified key presses by changing the position of its menuitem highlight. The author does not need to describe the behaviour associated with moving up and down these menuitems.

Components can also be specified to generate events to signal some internal change. This provides a mechanism to allow the service description to contain some custom behaviour beyond that defined in the component specification. For example, if a visual component receives the focus it will generate an "OnFocus" event, that could be used to drive some custom "roll-over" behaviour, so that contextual information is presented in a TextBox (somewhere else in the Scene).

The *Service* component contains both *Scene* components and any other components that are required to persist between these scenes. For example, an IntegerVar component used to store a high-score. Normally, the lifetime of all components within a scene component is the same as the lifetime of the scene itself. A transition from a scene will cause all child components and accompanying state to disappear. For this reason, any components that need to persist between scenes must be included in the service component.

4.3.1 Layout components

PCF provides for the layout of content in two ways – explicit layout and flow layout.

Explicit layout is described using the *ELC* (Explicit Layout Container) component. This component has no visual appearance of its own. It is instead a container for other components that need to be positioned at an exact location. The Service and Scene component are both ELCs.

An ELC component has an *origin* property, but no *size*. Components declared within an ELC component have their origins specified relative to the ELC's origin. Consequently, moving the parent ELC component's origin will cause all the child components to move by the same relative amount. Flow layout is described using a combination of *Flow* components and *flow layout container* components.

Flow components are used to manage the content to be presented, which may be a mixture of text and other PCF components.

- Flow layout container components are used to position the content described by a Flow component. This is achieved using "flow" rules, so that the content described by a Flow component is presented without requiring the service author to specify exact positions. For example, the Page Flow Container presents flowed content in a rectangular area of fixed width, height and origin. Content that exceeds the limits of this container will be flowed onto the next "page" and can be viewed by navigating through the multiple pages.

4.3.2 Custom components

The set of component classes standardised by DVB can be augmented by the creation of *custom components*.

A custom component can be created for a target platform to exploit platform capabilities that are not available through the use of standard PCF components alone. PCF service descriptions containing custom components will only work on target platforms that implement these components, and so are likely to be less portable than PCF services that use only standard components.

4.4 Content

Content is managed within a PCF service description using a set of standard PCF data types. In PCF's strongly typed model, all values are described with a data type so that it can be translated into the most appropriate type for a particular platform.

The PCF provides means for items to be grouped together and referenced in the form of structures and arrays. Structures, such as the Collection data type, provide a mechanism for related data items to be accessed as a single logical unit. Arrays allow multiple items of the same data type to be grouped so they can be easily indexed and iterated over.

4.5 Behaviour

All run-time behaviour that contributes to the viewer experience of a PCF service is event-driven.

When an **event** occurs it may be consumed by a component resulting in the execution of behaviour defined as part of that component's class specification. For example, the Menu component is specified such that it will react to the UP/DOWN user keys by moving its highlight up and down over its menuitems. The service author need not describe this behaviour.

Alternatively, the event may be consumed by a custom handler declared within the service description. This might be used to create custom behaviour that is not an inherent part of an available component's functionality, or for linking the behaviour of one component to that of another.

The PCF **action language** can describe sequences of actions to be executed either in direct response to an event occurring, or as a result of a state transition caused by an event. The following example includes an OnEvent, where some actions take place when a trigger condition is satisfied.

```
<Scene name="scene">
    <Collection name="variables">
        <IntegerVar name="counter">
            <Integer name="value" value="1"/>
        </IntegerVar>
    </Collection>
    <OnEvent>
        <Trigger eventtype="KeyEvent">
                <UserKey name="key" value="VK_ENTER"/>
        </Trigger>
        <ActionLanguage name="jumpCounter">
            variables.counter.value += 10;
        </ActionLanguage>
    </OnEvent>
</Scene>
```

The PCF supports the concept of **focus.** Focus effectively represents where the viewer's attention is currently focused, and so user input events are initially directed at the focused component. If the focused component is not interested then the event is propagated according to a set of event propagation rules.

4.6 PCF referencing mechanism

For a simple service it may be appropriate that all aspects of the service description are in one PCF source document. For larger services, however, it may not be feasible to locate the entire service description in a single PCF source document, for reasons of efficient description of authorial intent and/or the modularity required in some business-to-business interchange situations.

The PCF provides flexible referencing mechanisms to accommodate these extremes of service description structure.

The first method uses the **href** property. This enables an element to remotely include other PCF items. The value of this property describes a path to the item to be referenced, using the values of the *name* properties within the structural hierarchy of the service. This makes it possible to reuse of parts of a service. For example, a "boilerplate" set of components declared once can be included within multiple Scenes.

A second method of referencing uses the **copy** item. This enables the referencing of items contained within some other item. The copy item is effectively replaced with the items contained within the item it refers to.

For example, the collection named "MyContent" below contains two *string items*. The Scene declaration includes a Copy in order to "pull in" the content of the item called "MyContent" so that both *string items* are now described within the Scene. The TextBox components within the Scene each contain *string items* that use the *href* mechanism. In this instance, the href is used to identify individual pieces of text from "MyContent".

```
<Collection name="MyContent">
    <String name="Item1" value="Mozart"/>
    <String name="Item2" value="Beethoven"/>
</Collection>

<Scene name="Composers">
    <!-- Include referenced content within collection -->
    <Copy href="#../MyContent"/>

    <TextBox name="Box1">
        <!-- Refer to included content item -->
        <String name="content"  href="#../Item1"/>
        <Size name="size" value="100 40"/>
    </TextBox>

    <TextBox name="Box2">
        <!-- Refer to included content item -->
        <String name="content"  href="#../Item2"/>
        <Size name="size" value="100 40"/>
    </TextBox>
</Scene>
```

4.7 Profiles

Any service described using the PCF will rely on certain features being available on a target platform, such as a return path connection or support for complex video manipulation. The PCF cannot solve the problem that a feature required for a service may not be supported by a particular target platform. However, the PCF profiling mechanism provides an easy way of defining the minimum capabilities, i.e. the minimum profile, that a target platform must provide in order to

meet the authorial intent captured in the PCF description of an interactive service. This is so that an author can have confidence that their service description will work on a particular target platform, or more precisely a particular target device, without having to meticulously match up every feature they have used with the features provided by that target.

Typically, each PCF profile will embody a set of PCF features such that it maps well to a particular application area, very much as profiles have been used within MHP, e.g. enhanced broadcast, return path enabled, PVR enabled etc. In addition, to increase adoptability of the PCF, profiles have been designed to reflect steps in complexity of the PCF transcoder that is required.

4.8 Exchanging PCF assets

The PCF is intended primarily for business-to-business interchange, so it is important that it can be exchanged easily between different tools, content systems and networks. Although not mandatory, the PCF specification provides a business interface model that describes the encapsulation and workflow for this interchange. This means that systems implementing the model can, in addition to understanding a PCF service description itself, exchange the service's assets easily, in a way that guarantees service integrity.

To allow multiple PCF assets (PCF source documents and other content assets referenced by this source) within a service to be updated atomically, they can be grouped together into *transactions* for submission across the business interface. Each transaction carries a *priority* and this is to assist the receiving process in allocating resources to the task of processing the contents of the transaction. For example, if parts of a PCF service contains rapidly updating live sport scores, whilst another part of the service contains slowly updating, but sizeable, news stories, it is desirable to specify that the sport score content should be processed more quickly, and in preference to, the less latency-sensitive news content.

5 References

1. Cartwright, R., Hunter, J., Rosser, J. and Steele, T., 2005. Portable content format. DVB Scene. Vol. 14. pp.8.
2. Cartwright, R., Hunter, J., Rosser, J. and Steele, T., 2005. The DVB Portable Content Format – What is it all about? DVB World conference publication.
3. Cartwright, R., Hunter, J., Rosser, J. and Steele, T., 2005. Platform-independent business-to-business iTV interchange format. IBC conference publication.
4. Digital Video Broadcasting; Portable Content Format Specification 1.0, ETSI TS 102 523.