BBC Nitro API

An introductory guide

BBC Platform API

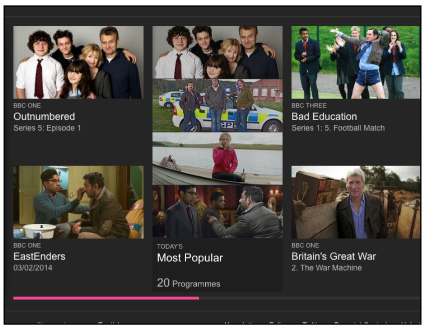
Revision History

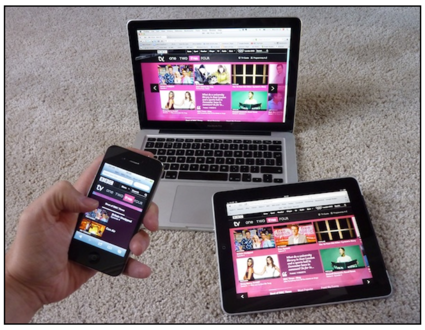
1.0-1.1 June, 2014: Internal draft releases
1.2 June 16, 2014: Details SLA on breaking changes, adds appendixes on titling and image recipes
1.3 August 8, 2014: Updates on Pulse
1.4 May 1, 2015: Updates to title logic and deprecation policies
1.5 June 17, 2015: Minor edit to the title logic

Note: This document describes an API under development. Details are believed to be correct at time of publication, but should be checked against the live system for correctness.

Introduction

SINCE THE 2007 launch of iPlayer, the BBC has been at the forefront of on-demand video and audio services. From what began as a PC-only desktop product, the BBC now makes its programmes available on demand to audiences on a wide range of devices and partner platforms.





Building metadata delivery systems to meet these new requirements has proved a significant challenge, and in response the BBC has embarked upon a multi-year project to replace legacy systems and databases with a performant, scalable platform for the future.

A key component of this new platform for accessing real-time programmes metadata is the Nitro API. Already used significantly inside the BBC, including for the new iPlayer and the Winter Olympics, we are now ready to offer BBC Nitro to partners and the general public.

Nitro is a major upgrade to our former offerings, including the MRSS feeds. Benefits include:

- You use the same APIs as BBC products (including iPlayer) which means features and content are available to you at exactly the same time as they are available for iPlayer
- You are not restricted only to programmes made available via broadcast, as you were with previous APIs: Nitro exposes the full range of content, including iPlayer Premieres and supports the forthcoming 30 Days of online availability as standard.
- Nitro is actively developed and fully supported

The guide is designed to help those interested in using BBC Metadata get started with Nitro, and assumes no prior knowledge.

Overview

Via BBC Nitro, the BBC will makes available XML and JSON feeds providing programme and schedule metadata. The format of the feeds is now custom to the unique requirements of BBC clients, and documented in a schema which may change from time to time.

The feeds are the same as those used by BBC products, including the iPlayer, and provide:

- Linear broadcast schedules (including regional variations) for all 10 BBC TV channels and 59 radio stations, up to 7 days in the future, and with details of On Demand availability
- A catalogue of all BBC programmes available on demand, filtered by categories
- Collections of programmes curated by BBC editorial, including “editors’ highlights”
- Guidance information for programmes according to the Denton code scheme
- Images suitable for display in EPG (subject to licence conditions)

The Nitro Way

Previous BBC metadata APIs included multiple elements that are expensive and slow to compute. By contrast, with Nitro feeds are much smaller and more focused, meaning they can be delivered faster and more efficiently. However, as a result clients are expected to make multiple calls to get all the metadata they need.

Some other major conceptual differences include:

Nitro Feeds are lists of entities

The B2B feeds are often modelled around tasks, for instance "find the contents of a container". Nitro differs from this. Nitro feeds can all be thought of as lists of entities, which you then filter and sort to reduce to just those items you care about.

In Nitro:

- **Feeds** are the building blocks, and offer lists of entities. The Schedule feed, for instance, offers a (paginated) list of all the broadcasts made by the BBC.
- **Filters** are means of reducing that list to just those entities you care about. So you may add the "only the BBC One service" and "tomorrow" filters to the schedules feed just to get the subset of broadcasts on BBC 1 tomorrow.
- **Sorts** help to manage the lists. A "views" sort on Programmes gives you episodes in order of their popularity, for example, while a "title" sort applies editorially-aware title sorting to the list.
- "**Mixins**" allow you to add optional decoration to results, at the expense of slightly slower performance. This allows us to keep basic calls fast while saving you unnecessary additional calls for common use-cases. For instance, you can mixin the full cast and crew information for a programme.

Availability

On Demand availability is exposed as another filter for Nitro feeds. Where supported, you can filter results to show only those which are available On Demand (or coming soon within a given time period). You must supply a media_set to specify which platform you are querying for availability on.

Details of media_sets are available separately from the BBC. (If you just want a general guide to desktop availability, use the pc media set: media_set=pc)

Using Nitro

With Nitro, we intend for the API to be as self-documenting as possible. Part of this can be seen with our use of 'hypermedia', where the API responses contain within them the documentation of available parameters and links to further queries. This includes full details of the available feeds, filters and sorts.

It's important that you consume these programmatically where possible as it is part of your responsibility as a client to be aware of changes to the API and respond accordingly (see below for more on client responsibilities). A schema for Nitro responses is also available, at /nitro/api/schema in all Nitro environments.

Paging

Nitro results are paged for performance reasons. You can specify the number of results you desire in a page with the page_size parameter, however we reserve the right to return only the number of results we can support in a performant way. Maximum page sizes are subject to change at any time, based on BBC operational requirements.

The paging can be seen in each response. In the XML view, it currently looks like:

```
<nitro>
        <pagination>
                <previous href="/nitro/api/programmes?page=1" />
                <next href="/nitro/api/programmes?page=3" />
        </pagination>
        <results page="2" page_size="10" total="2148977" more_than="“2148976">
```

Some important points to note here:

- To move through pages, you should follow the previous or next hrefs given in the response. This will ensure you get the right results even if the page size changes from the one you requested.
- Note the page_size attribute of the <results> element. This tells you how many results Nitro is currently returning per page, and may not be the same number you asked for.
- The total attribute is not guaranteed to be present, as it can be expensive to compute. more_than will always be present, but the actual number of results can be substantially higher than this figure in some cases.

API Keys

Access to Nitro is governed by the use of an API key. You register for a key on our developer site at http://developer.bbc.co.uk/, and should include it with every request via the api_key parameter. This key will govern your rate limits; if you need to review these please contact nitro@bbc.co.uk.

If your key exceeds its rate limitation you'll be given an HTTP 500 error. The body of response will be similar to:

```
{ "fault": {"faultstring": "Rate limit quota violation. Quota limit : 0 exceeded by 1. Total violation count : 1. Identifier : YOUR-API-KEY-HERE",
   "detail": {"errorcode": “policies.ratelimit.QuotaViolation"}}}
```

Connecting to Nitro

Nitro’s “front page” is available at /nitro/api on all environments. That page is the starting point for Nitro discovery, and includes a description of all feeds, filters, sorts and mixins. The current host for Nitro is nitro.api.bbci.co.uk. Once you have an API key, get started at:

http://nitro.api.bbci.co.uk/nitro/api/?api_key=YOUR-KEY

A test environment is also available. Contact the BBC if you would like access to this.

Caching

Nitro responses include cache control headers, and it is a requirement that you honour these.

Content Negotiation

To specify whether you want JSON or XML responses, set your accept: header to either application/xml or application/json respectively.

Dates and Times

Dates and times are given in ISO 8601 format. For example, 19:28:00 on 17th March 2014 is shown as 2014-03-17T19:28:00Z. Datetimes are in UTC in all elements, including for broadcasts.

Durations are also given in the ISO 8601 format. For example, 30 minutes, 25 seconds is shown as: PT30M25S

Images

Where available, programmes will contain an image, for example:
<image pid="p01m62fc" template_url="http://ichef.bbci.co.uk/images/ic/$recipe/p01m62fc.jpg" /> You should not assume any specific resolution, and should scale the image appropriately for the output system by replacing the $recipe placeholder in the template_url with your desired size - eg 256x144 to give http://ichef.bbci.co.uk/images/ic/256x144/p01m62fc.jpg. Supported resolutions will be supplied by your BBC contact.

Important: end-user clients (eg the set-top box itself) should not request images directly from BBC servers. Your metadata agreement contract will give more detailed restrictions on the use and manipulation of BBC images.

Nitro client responsibilities

Rate limiting

As mentioned above, Nitro is rate limited for all clients, and you should consider how your client will behave in the event of a rate limitation. We do this to ensure no one client consumes all of the available Nitro resource to the exclusion of others, as has happened in the past.

- If your volumetrics expect a high number of calls to your product or service, you **should** consider a pattern that manages how many of those requests turn into unique Nitro calls
- If you need a higher rate limit than that available, you may need to request one from the BBC. Contractual partners should approach their BBC representative in the first instance.
- You **should** behave appropriately if Nitro fails to respond for short periods or responds with a non-200 status.

Clients **must avoid** retrying failed requests indefinitely, and should use random delays after a failure. Some clients have chosen to implement circuit-breakers to help with this.

Responding to changes

Nitro feeds are continually evolving and subject to change. While we don't break existing calls, we **do** deprecate them and replace them with better alternatives as they become available.

One of the major challenges in evolving our previous metadata systems were the problems encountered trying to deprecate old features or change them to newer and better ways of doing things. That means that over time our systems carried multiple redundant versions of feeds, and feeds grew enormously in size as new features were added and old ones could not be removed. By asking clients to enter into this contract with us, we hope to keep Nitro lean and improve it continually.

We indicate the changes to clients with a variety of attributes in our hypermedia. The best place to see all the hypermedia is the "front page" - /nitro/api/. There you can see, for example:

<sort name="most_popular" is_default="false" title="sort numerically by popularity (most popular first)" release_status="**deprecated**" deprecated="true" deprecated_since="2013-11-11" replaced_by="views">

The most important one to watch is release_status. Possible values are:

- **Supported**: This is a fully supported Nitro feature, and has the full history of data available. You can use this feature in production code
- **Beta**: This feature has been developed, and is available for you to preview and build against. It may not have the full history of data available, and you use it in production code at your own risk. Responses are subject to change.
- **Alpha**: This feature is under development, and should not be used in production code. It is highly likely to change, or not operate as expected.
- **Deprecated**: This feature is scheduled for removal and should be removed from production code as soon as possible. If a replacement is available, it will be listed under

replaced_by, although the replacement is not guaranteed to have identical responses to the previous feature.

Once we mark something as deprecated, we mark it with a date under "deprecated_since" and it is listed in the `<deprecations>` block at the end of relevant Nitro responses. Actually removing the feature is done by a timetable set by the BBC, and we guarantee that this will not be any faster than **three months** from the time of deprecation.

Nitro is backward compatible in the sense that we guarantee not to remove or change existing features without first deprecating them (except in the case of bug fixes). We expect you to be forward compatible in that you must ignore unknown elements and attributes, and assume no ordering beyond the first level within the results element. Our published XML schema — /nitro/api/schema/ — can be expected to change in accordance with these requirements.

- Clients are **required** to be aware of deprecation changes and respond to them appropriately.
- You **must** be forward compatible, and not make unwarranted assumptions about the content or shape of Nitro. We make backward-compatible changes to Nitro frequently and your code must be able to accept this.
- From experience with other clients, we **recommend** you do not hard-code paths to calls in your code. What is ?sort=most_popular can become ?sort=views and you don't want to have to change this in hundreds of places. Instead, try to follow the hrefs we provide in our hypermedia. Follow those and you'll become partly insulated to this problem.
- You **must not** dissect the hrefs or assume the availability of filters, sorts, or mixins outside those that are published in the hypermedia.
- You **should** respond to the content of the response and not hard-code assumptions that can be found in the Nitro output. (EG, do not write code that fails catastrophically if the returned page_size differs from the requested page_size)

Calling patterns

Nitro feeds are not richly decorated, but are instead focused on the minimum required to fulfill the majority of clients' needs. That means, for instance, the programmes feed returns a good minimum of detail for an episode, but if you want broadcast details for a particular episode you will have to make an additional call to the Schedules feed, or if you want details of chapter points you will have to make an additional call to the Items feed.

This design decision on our part implies that for many client use-cases, fully decorating a programme or other entity may require multiple calls to Nitro. To try and help bridge the gap, we also offer "mixins" on some Nitro feeds.

These do offer the rich decoration, but come at the expense of performance. When using a mixin, we make no guarantees around performance. In particular, clients should not expect calls with mixins enabled to return in under 300ms.

Nitro Pulse

What is Pulse?

Nitro is designed on the whole to be a real-time interactive system: a user asks for a page (eg the iPlayer home page) and Nitro finds and supplies currently available programmes to fill it with. However, for many Nitro clients they are interested in a different pattern, one where they fill/sync their own separate databases with information from Nitro.

Syncing information with Nitro has previously been difficult: you have to page through the entire set of results for a filter (which can be extensive) and then repeat the exercise on a schedule, even for pages which haven't changed since you last asked. This puts unwanted load on BBC systems, and also results in clients exceeding their rate limits when performing what are essentially redundant operations.

Nitro Pulse is an attempt to resolve these problems by offering clients a "change feed" that gives them only the differences in the feed results since they last asked.

How does it work?

With Pulse, each set of Nitro results includes a changes href at the bottom. This includes a token that you can include on your next request to Nitro. If you include the token, instead of being given the full set of results, you will get only those entities which have been added, removed or changed since you last asked. For example:

1. You ask for available Eastenders clips:
/nitro/api/programmes?descendants_of=b006m86d&entity_type=clip&availability=available&media_set=pc
2. You get more than 1500 results. On the first page, you grab and store the changes href.
3. Page through the results and ingest them into your system.
4. When you're finished and ready to ingest the changes, call the href you stored in step 2. It's crucial to use the exact href — even if you see what looks like a datetime, don't reverse-engineer this and provide a date of your own choosing.
5. The feed will now supply only the differences since your first original call to Page 1. For each change, the full standard entity will appear but with an additional change_type element, which can have three values:
    1. Added: the entity has been added to the results. In the example case, this means a new clip has now become available
    2. Deleted: the entity has been removed from the results. In the example case, the clip is no longer available (it may have been revoked, for instance)
    3. Changed: the entity differs from the one you were supplied originally. For instance, in the example case its title may have been updated, or it may have a different synopsis

Are there push notifications of changes?

Yes, a second part of Pulse supports change notifications via an AWS message queue.

When I can use Pulse?

Pulse is not yet available for general use. Please contact Nitro team for further information.

Frequently Asked Questions

How do I get details for an episode of a programme?

If you have the PID, say b038l3g1, go to the Programmes feed directly: /nitro/api/programmes?pid=b038l3g1. This will give you all the details on your programme, including images, genres and more.

If you don't have the PID but have a parent brand or series (eg, for Eastenders, that's b006m86d) use the children_of filter: /nitro/api/programmes?children_of=b006m86d. This finds the child programme(s).

Want to exclude clips from this search? Add entity_type=episode - /nitro/api/programmes?children_of=b006m86d&entity_type=episode

Get only available episodes of a programme?

The above query returns 1500+ episodes of Eastenders. To narrow this to just those that are available to watch online, use the availability filters. Some notes here:

- You'll need a media_set to specify which platform you're checking for availability. If in doubt, pc is a reasonable default, but includes some content unavailable on mobile platforms.
- You can currently specify availability=available to filter for only those programmes currently available, availability=pending for those that are very imminently about to become available, or supply an ISO-8601 duration (eg PT5H) to filter for those that will be available within the next five hours. Note you can also combine these.
- Due to the PIPs model, availability "bubbles up" to parents in the hierarchy: the Eastenders brand is considered available if any of its child episodes are available. This leads to a situation when an episode is not available to watch in its entirety but shows as available because it has child clips that are available. To avoid this, set availability_entity_type to episode, which excludes clips from the calculations.

Find the "parents" of a programme?

If you're only interested in the title and the pids for any parents of a programme, use the ancestor_titles mixin to insert this into the programmes or schedules feeds.

Build a Now/Next for a channel?

For PIPS, Broadcasts (transmissions) don't happen on a channel or network, they happen for a service. To build a schedule or work out what's on, you need to choose the service you need. For BBC One, the usual default is to use London -- bbc_one_london, to PIPs.

Ask the Schedules feed for the first programme on your service that has not yet ended, and the programme after that, plus (optionally) their titles. Here's how.

Filter schedules for BBC One London schedules?sid=bbc_one_london and only for the next ending programme (assume it's 5pm GMT on Aug 15 when we make this call) &end_from=2013-08-15T17:01:00Z then limit to two results for now and next &page_size=2 and mixin in the titles from the Programmes feed to save me making a second call &mixin=ancestor_titles

Get a title for a programme?

Titling logic can be involved, as it often means combining the Brand and Series titles with the Episode titles. Certain clients, including those with commercial syndication agreements with the BBC, are required to follow the logic used by iPlayer, which is detailed in Appendix 3 below.

To help you construct a title, the mixin=ancestor_titles option on /Programmes or /Schedules and Nitro will walk the PIPs tree and find you as many titles as it can. Some episodes also include a Presentation Title, though it may not in practice be the most useful.

Note too that in PIPs, a Brand or Series are not guaranteed to exist (or even to have titles, if they do). See the entry on TLEO below for more details here.

Find the broadcast times for a programme?

If you know the programme pid, eg b01mxx3h for a given Doctor Who episode, use the schedules feed: /nitro/api/schedules?descendants_of=b01mxx3h. Note that in this case you'll get 22 results. That doesn't mean it was shown 22 separate times, only that it was shown once across the 22 BBC One regional Services.

From the results, look at the published_time element's start and end attributes to find its scheduled times. Where we have specific live times available, we will include a tx_time element with an extremely precise start/end.

Do a free-text search on Nitro feeds?

The q parameter where supported allows free-text search against all the metadata in a document. Use the title: or synopses: keywords with the filter to only search against those elements.

What Is ...?

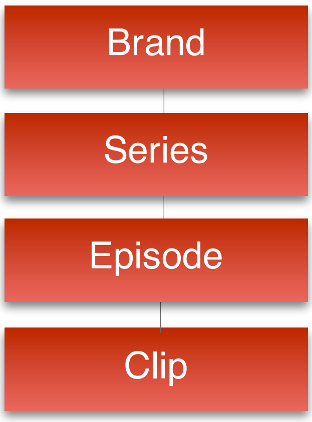
PIPS?

PIPs is the coverall name for the Programme Information database(s) and their associated services and APIs. Nitro is the new read API for the latest version of PIPs, PIPS 4. PIPs contains editorial objects like programmes, people, items (tracklists and chapters), as well as technical objects like media assets, right contracts and so on.

A PID?

A PID is a Pips ID. Everything in PIPs has a PID. This means that you don't know what a PID represents until you try it. You'll often hear PIDs referred to by the type of thing they are - "a programme PID", "an episode PID", "a vPID" (version pid) and so on.

A TLEO?



The TLEO is the Top Level Editorial Object. Programmes in PIPs are stored in a tree: (see right). Brand is the top of the tree, then series (and sub-series if any), episode, and clip.

None of the levels of the tree are guaranteed to exist: A clip can stand alone; A brand can have no episodes; an episode can have no series. Some examples might help here:

- Eastenders episodes are all children of a single Eastenders brand, as Eastenders has no concept of a series. (In general, PIPS does not create a series until there is a second series that needs to be distinguished.)
- Films are the usual example of episodes that have no series and no brand
- A series can have another series as child. This is how PIPs represents two-part episodes.

A Programme?

To Nitro a Programme is anything on the programmes tree: a brand, a series, an episode or a clip.

A Broadcast?

It is important to distinguish between the programmes themselves (the episodes, for instance) and their publication events. This is when a programme is transmitted to an audience. In the classic model, this is a broadcast to a TV or radio, but publication events include webcasts (when programmes are solely streamed live online).

All of these publication events have PIDs as well as their programmes.

The two worlds of programmes (the editorial objects) and publication events (broadcasts) meet in the Version entity.

A version is an editorial cut of a programme - eg the original cut, or a version with sign-language, or an audio-described version, or a version cut for language.

A publication event is always of a version, not a programme.

An example: The programme with PID b01mxx3h is an Episode of Doctor Who Series 7, titled "A Town Called Mercy". See it in Nitro here: /nitro/api/programmes?pid=b01mxx3h. This is the editorial programme. It has an Original version, pid b01mxx1y, which you can see at /nitro/api/versions?pid=b01mxx1y.

That original version was broadcast on BBC One London at 18:35 GMT (All Nitro times are GMT) on 15th September 2012. That broadcast had a pid, p00y97y2 /nitro/api/broadcasts?pid=p00y97y2.

The key thing to note here is what an audience member might think of as a "programme" has (at least) three pids: b01mxx1y, b01mxx3h, and p00y97y2. The first is the programme PID and that is nearly always what you want to use. The version pid is too specific, and the broadcast pid only refers to a single broadcast.

And note that you can't tell what type of thing a PID is just by looking at it.

A Service?

For PIPS, a service is the means by which publication events happen - it's what broadcasts are broadcast on. What we think of as channels or networks (like BBC One or Radio 4) are comprised of multiple services – BBC One London, BBC One Oxford, BBC Radio 4 Longwave and so on.

Nitro's services feed helps you find the services you need to use for filtering broadcasts./nitro/api/services?service_type=TV is the list of all TV services.

Additionally, there are services that drive OnDemands and Simulcasts for online playback. These are generally only used in media playback, and are not generally useful for finding schedule information.

Appendix 1: Nitro feeds and the B2B feeds

As an assistance to users of the deprecated MRSS/B2B feeds, below is a guide to the Nitro feed that allows you to get similar functionality:

The Schedule Feeds

For a given BBC service, e.g. BBC One London, you use the Schedule feed to obtain a list of broadcasts. Use the schedule_day filter to obtain just those broadcasts for a given BBC schedule day (6am London time - 5.59am London time).

Note: The B2B schedule feed accepts a media_set here, which is used to supply details of OnDemand availability. Nitro does not support this, as it is expensive to compute. Instead, use the Programme feeds (details below) to obtain this information.

The Container Feed

The B2B "Container" feed was used to generalise the BBC concepts of "series" and a "brand". In Nitro, the BBC programme hierarchy is grouped together in the Programmes feed. Use the descendants_of filter on the programmes feed to find all the members of a B2B "collection", optionally filtered by their availability.

The Episode Feed

To find the details of an individual episode from Nitro, the starting point is the programmes feed. Use the pid filter to specify which episode you are interested in.

Additional information that is not part of the basic feed, such as availability window information, can be found via mixins (in this case the availability mixin).

The Masterbrand Feeds

The B2B Masterbrand feed provided all programmes available on-demand episodes for a given masterbrand.

In Nitro, the same result is achieved by filtering the programmes feed with the master_brand filter and specifying entity_type=episode and then applying availability and media_set filters as usual.

The Category Feeds

Similarly to the Masterbrand feed, in Nitro this feed is served from programmes. Give your BBC-supplied category code to the genre filter, and apply availability and media_set filters if required.

The Mostpopular Feeds

This feed, which supplies the most popular available episodes, has been replaced in Nitro with a sort on the programmes feed.

You can now sort any query to programmes by views, which will return items in the order of their popularity in the last 24 hours.

The Featured Feeds

To obtain the list of editorially chosen episodes, use the promotions feed, with the filter promoted_for. You will be given a value to supply by your BBC contact.

This will return you episode PIDs, which can be further decorated with calls to the programmes feed as required.

The Collection Feeds

From time to time the BBC will supply partners with “collection” IDs, which are for editorially curated collections of programmes (eg “Glastonbury”, “World War One”). These IDs should be used with the group filter on programmes to get the list of programme members of the collection.

Linear service identifers, masterbrands and categories

Service identifiers are available from the services feed. Masterbrand details are available from the master_brands feed. Category genres you should use will be supplied to you by your BBC contact, and may change from time to time.

Appendix 2: Editorial Requirements

Clients who use Nitro under a syndication agreement with the BBC must meet BBC requirements surrounding the editorial presentation of metadata (and these are best practice for all Nitro users). Below is a brief guide to those that can be answered with calls to Nitro

Branding

- Must display channel attribution wherever BBC content is represented.
- Must use the correct channel names, not service names (BBC One, not BBC1)
- Should use the channel block logos

To get channel attribution, use the master_brand element of Nitro responses, e.g.:

```
<master_brand result_type="master_brand" mid="bbc_one"
href="/nitro/api/master_brands?mid=bbc_one" />
```

By following the href you will get the image and channel name required for proper attribution.

Programme Information

- Must display Title and Subtitle
- Must display Synopsis in full
- Must display programme image

These first three are available from the Nitro programmes feed. For details on constructing a title, see Appendix 3. Synopsis and Image are available directly from the feed. (There are three lengths of synopsis provided for you to choose whichever best meets your requirements)

- Should show programme broadcast time

This comes from a call to the Schedules feed: Use the descendants_of filter with your episode pid, sort by scheduled_start and give a page_size of 1 to get the first broadcast.

- Must display parental guidance messages
- Should show programme length

Both parental guidance messages and programme length (shown as an ISO-8601 duration) are properties of the version of an episode. To find versions of an episode, use the descendants_of filter on the Versions feed. Eg, extracted from /nitro/api/versions?descendants_of=p0206m53:

```
<duration>PT1H</duration>
<warnings>
    <warning_text length="short">Contains some violence.</warning_text>
    <warning_text length="long">Contains some violent scenes.</warning_text>
    <warning short_description="some violence" warning_code="V1">contains some
    violent scenes</warning>
</warnings>
```

Appendix 3: Creating a Title

To construct the correct title for a programme, you must consider the multiple possible levels of the PIPs programme hierarchy. As mentioned above, some episodes have both a series and a brand as parents, some only have a brand as parent and some have neither.

Clients using Nitro under a BBC syndication agreement are required to follow these steps to construct a title; it is provided to others as a convenience.

To determine the title:

1. If there is a brand title in the ancestry, use the brand title
2. If there is no brand title in the ancestry, but there is a series, use the first series title
3. If there are no ancestors for the episode, use the episode title
4. If there is no title for the episode, use the presentation_title

To determine the first part of the subtitle:

1. If the episode has no ancestors, there is no first part of the subtitle
2. If the episode has a series and a sub-series in its ancestry, use the sub-series title
3. If the episode has both a series and brand in its ancestry, use the series title

To determine the second part of the subtitle:

1. If the episode has no ancestors, there is no second part of the subtitle.
2. Otherwise, the second part of the subtitle should begin, ": ", then the episode position, then a full stop, then the episode title. EXCEPTIONS:
   1. Do not include the ": " delineator if there is no first part of the subtitle
   2. If there is no episode title, use the episode presentation_title
   3. Do not include the position where the episode title being used (either title or presentation_title) begins with any of these structures:
      - "Episode n" where n is an integer
      - "Part n" where n is an integer
      - A date format (eg "dd/mm/yyyy")
      - "Week n" where n is an integer

To get the titles for the ancestors of an episode, you may use the ancestor_titles mixin.

Example: For pid p01jw8tk the title should be "Moving On" with the subtitle "Series 5: 5. Back By Six". An extract of /nitro/api/programmes?pid=p01jw8tk&mixin=ancestor_titles is:

```
<episode><pid>p01jw8tk</pid>
    <title>Back By Six</title>
    <presentation_title>Episode 5</presentation_title>
    <episode_of result_type="series" pid="p01jw77y" position="5"
    href="/nitro/api/programmes?pid=p01jw77y" />
    <ancestor_titles>
        <brand><pid>b00vt2q1</pid> <title>Moving On</title></brand>
        <series><pid>p01jw77y</pid><title>Series 5</title></series>
    </ancestor_titles>
</episode>
```

Appendix 4: Image recipes

Nitro provides image references as a path to the image which includes a placeholder for a "recipe". Supported sizes are listed below. EG:

<image pid="p01lt65t" template_url="http://ichef.bbci.co.uk/images/ic/$recipe/p01lt65t.jpg" />

Replace "$recipe" a code from below: http://ichef.bbci.co.uk/images/ic/128x72/p01lt65t.jpg

Width	Square	16:9	Native
16px	16x16	16x9	16xn
32px	32x32	32x18	32xn
48px	48x48	48x27	48xn
64px	64x64	64x36	64xn
80px	80x80	80x45	80xn
96px	96x96	96x54	96xn
112px	112x112	112x63	112xn
128px	128x128	128x72	128xn
144px	144x144	144x81	144xn
160px	160x160	160x90	160xn
176px	176x176	176x99	176xn
192px	192x192	192x108	192xn
208px	208x208	208x117	208xn
224px	224x224	224x126	224xn
240px	240x240	240x135	240xn
256px	256x256	256x144	256xn
272px	272x272	272x153	272xn
288px	288x288	288x162	288xn
304px	304x304	304x171	304xn
320px	320x320	320x180	320xn
336px	336x336	336x189	336xn
352px	352x352	352x198	352xn
368px	368x368	368x207	368xn
384px	384x384	384x216	384xn
400px	400x400	400x225	400xn
416px	416x416	416x234	416xn
432px	432x432	432x243	432xn
448px	448x448	448x252	448xn
464px	464x464	464x261	464xn
480px	480x480	480x270	480xn
496px	496x496	496x279	496xn
512px	512x512	512x288	512xn
528px	528x528	528x297	528xn
544px	544x544	544x306	544xn
560px	560x560	560x315	560xn
576px	576x576	576x324	576xn
592px	592x592	592x333	592xn
608px	608x608	608x342	608xn
624px	624x624	624x351	624xn
640px	640x640	640x360	640xn
656px	656x656	656x369	656xn
672px	672x672	672x378	672xn
688px	688x688	688x387	688xn
704px	704x704	704x396	704xn
720px	720x720	720x396	720xn
736px	736x736	736x414	736xn
752px	752x752	752x423	752xn
768px	768x768	768x432	768xn
784px	784x784	784x441	784xn
800px	800x800	800x450	800xn
816px	816x816	816x459	816xn
832px	832x832	832x468	832xn
848px	848x848	848x477	848xn
864px	864x864	864x486	864xn
880px	880x880	880x495	880xn
896px	896x896	896x504	896xn
912px	912x912	912x513	912xn
928px	928x928	928x522	928xn
944px	944x944	944x531	944xn
960px	960x960	960x540	960xn
976px	976x976	976x549	976xn
992px	992x992	992x558	992xn
1008px	1008x1008	1008x567	1008xn
1024px	1024x1024	1024x576	1024xn
1040px	1040x1040	1040x585	1040xn
1056px	1056x1056	1056x594	1056xn
1072px	1072x1072	1072x603	1072xn
1088px	1088x1088	1088x612	1088xn
1104px	1104x1104	1104x621	1104xn
1120px	1120x1120	1120x630	1120xn
1136px	1136x1136	1136x639	1136xn
1152px	1152x1152	1152x648	1152xn
1168px	1168x1168	1168x657	1168xn
1184px	1184x1184	1184x666	1184xn
1200px	1200x1200	1200x675	1200xn
1216px	1216x1216	1216x684	1216xn
1232px	1232x1232	1232x693	1232xn
1248px	1248x1248	1248x702	1248xn