Especificação da REST API do Repositório de Aplicativos para TV Digital

Índice


Ministério das
Comunicações
GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA


RNP

Patch

CHANGELOG

18 de Janeiro de 2016 - Dynavideo

- Aprimoramento da resposta na seção Avaliação do aplicativo
- Adição da chamada Atualizar uma aplicação por faixa através de patches
- Aprimoramento da Estrutura do arquivo PATCH para suportar atualizações por faixa através de patch
- Atualização da seção Atualização das aplicações via Internet -> Patch
- Atualização da seção Atualização das aplicações via sinal de TV Digital -> Patch

2 de Dezembro de 2015 - Dynavideo

- Adição do CHANGELOG
- Alteração do campo appIcon na seção Estrutura do arquivo MANIFEST
- Alteração do campo appEntryPoint na seção Estrutura do arquivo MANIFEST
- Alteração do campo appSigneFiles na seção Estrutura do arquivo MANIFEST
- Alteração do campo patchSigneFiles na seção Estrutura do arquivo PATCH
- Alteração do campo diffs na seção Estrutura do arquivo PATCH
- Alteração do campo appIcon na seção Estrutura do arquivo PATCH
- Alteração do campo appEntryPoint na seção Estrutura do arquivo PATCH
- Alteração do passo-a-passo de atualização via PATCH na seção Atualização via PATCH
- Melhoria dos exemplos
- Atualizaçao do Índice

16 de Novembro de 2015 - Dynavideo

- Versão inicial do documento

Introdução

O presente documento trata da especificação de uma API REST para o **Repositório de Aplicativos para TV Digital** do governo federal. Através dessa API será possível listar, buscar e recuperar informações sobre os aplicativos cadastrados no repositório. O objetivo do Repositório de Aplicativos para TV Digital é prover uma plataforma pública e gratuita que possibilite: armazenamento, intercâmbio, recuperação via streaming de dados sob demanda e execução de aplicações de TV Digital Interativa em conformidade com o middleware Ginga. O **Repositório de Aplicativos para TV Digital** está acessível através de um portal, ver Figura 1, e que pode ser acessado via seguinte URL:

http://gingabr.comunicacoes.gov.br

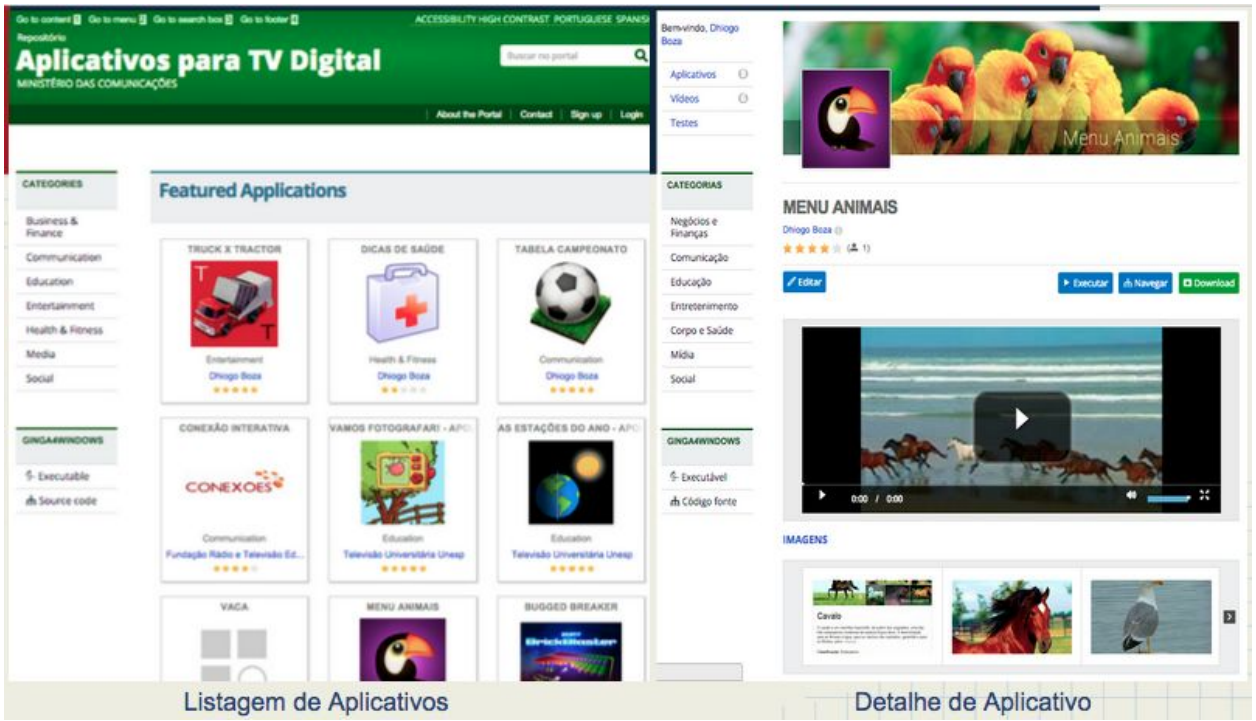

Figura 1 - Screenshots do Repositório de Aplicativos para TV Digital

Especificação da API REST

Models

Aplicação

Atributo: tipo	**Descrição**
id: long	Identificador único da aplicação.
name: string	Nome da aplicação.
promotionalText: string	Pequeno texto destinado a chamar a atenção do usuário. Contém um discurso conciso do objetivo da aplicação.
file: long	Identificador do arquivo com a fonte.
fileSize: long	Tamanho do arquivo do aplicativo descompactado (em KB).
rating: int	A avaliação de cada aplicativo (a quantidade de estrelas); [0 - 5]
userRating: int	Avaliação do usuário
parentalControl: int	Classificação indicativa do aplicativo, pode ser: 0: Livre 1: 10 anos 2: 12 anos 3: 14 anos 4: 16 anos 5: 18 anos
iconUrl: string	URL do ícone da applicação
coverUrl: string	URL da cover da applicação
category: int	Categoria do aplicativo, pode ser: 1: Negócios e Finanças 2: Comunicação 3: Educação 4: Entretenimento 5: Corpo e Saúde 6: Mídia

	7: Social
highlights: bool	true: Destaque false: Normal
screenshots: long[]	Array dos ids dos screenshots do aplicativo.
developerId: long	Id do desenvolvedor da aplicação.
developerName: string	Nome do desenvolvedor da aplicação.
controlCode: int	Código de controle da aplicação, pode ser: 0: autostart (começa imediatamente) 1: present (usuário precisa iniciar) 2: unbound (aplicação persistida) 3: kill (mata as aplicações que estejam sendo executadas)
lastChanges: string	Descrição das últimas modificações.
description: string	Descrição detalhada da aplicação.
version: string	Versão do aplicativo.

Desenvolvedor

Atributo: tipo	**Descrição**
id: long	Identificador único do desenvolvedor
name: string	Nome do desenvolvedor
email: string	E-mail para contato
website: string	Website

Chamadas

Categorias (GET /rest/v1.1/categories/)

Resposta:
200 - Ok

500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/rest/v1.1/categories/
Host: gingabr.comunicacoes.gov.br
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT

{
   “categories”: [
       {
           “id”: 1,
           “name”: “Negócios e Finanças“
       },
       {
           “id”: 2,
           “name”: “Comunicação“
       },
       {
           “id”: 3,
           “name”: “Educação“
       },
       {
           “id”: 4,
           “name”: “Entretenimento“
       },
       {
           “id”: 5,
           “name”: “Corpo e Saúde“
       },
       {
           “id”: 6,
           “name”: “Mídia“
       },
       {
```

```
            "id": 7,
            "name": "Social"
        }
    ]
}
```

Procura (GET /rest/v1.1/search/{q})

Requisição:

Parametro: tipo	**Descrição**
q: string [opcional]	Termo buscado.
rows: int	Número máximo de resultado por página. Caso ausente, serão retornados no máximo 10 resultados por página.
page: int	Número da página a ser retornada, começando de 1. Determina o deslocamento dos resultados exibidos. Caso ausente, será retornada a primeira página de resultados. Caso maior que o numero total de páginas, será retornada a última página.
developer: long [opcional]	Se presente, restringirá a busca a aplicações desenvolvidas pelo desenvolvedor passado. Se o id passado for invalido, não será retornado nenhum resultado.
category: int [opcional]	Restringe a busca a aplicativos da categoria passada. A categoria passada pode ser: 1: Negócios e Finanças 2: Comunicação 3: Educação 4: Entretenimento 5: Corpo e Saúde 6: Mídia 7: Social
highlights: bool [opcional]	Restringe a busca a aplicativos em destaque.
order: int [opcional]	Ordenação do resultado da busca. Se não especificado o padrão é utilizado. 0: Mais recente (padrão)

	1: Mais antigo 2: A-Z 3: Z-A 4: Mais visualizado 5: Melhor avaliado 6: Mais baixado
macAddress: string [opcional]	Endereço MAC do dispositivo, se presente e se a aplicações tiver sido avaliada através desse MAC, o campo userRating estará presente

Resposta:

200 - Ok

412 - O parâmetro `q` está faltando

500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/rest/v1.1/search/foto?rows=50&page=2&category=3 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT

{
  “count”: 95,
  “pages”: 5,
  “apps”: [
    ...
    {
      “id”: 5555555,
      “name”: “Vamos Fotografar! - Apolônio e Azulão”,
      “promotionalText”: “Ajude o Azulão a tirar fotos das plantas que dão flores e das
plantas que dão frutos! Mova a câmera com as setas do controle remoto e tire uma foto com o
botão "OK".”,
      “file”: 1761442,
      “fileSize”: 321854,
      “rating”: 3,
      “userRating”: 4,
      “parentalControl“: 0,
      “iconUrl” : “http://gingabr.comunicacoes.gov.br....”,
```

```
      "coverUrl" : "http://gingabr.comunicacoes.gov.br....",
      "category": 3,
      "screenshots": [9, 656, 1221],
      "developerId": 123456,
      "developerName": "Televisão universitária da Unesp",
      "controlCode": 0,
      "lastChanges": "31/05/2014: Envio da aplicação.",
      "description": "Apolônio e Azulão é uma série de miniprogramas voltados ao ensino dos
mais variados tópicos ao público infantil. Os episódios tratam especialmente de curiosidades
sobre o mundo que nos cerca e sobre seu funcionamento. A série gira em torno de dois amigos:
o Professor Apolônio, um simpático e estudioso velhinho que se diverte em explicar os temas
que surgem durante suas aventuras, e Azulão, o alegre, agitado e observador ser azul que
vira-e-mexe traz curiosidades do seu variado cotidiano quando se encontra com o professor.
Os encontros desses dois personagens geram histórias que partem de momentos simples e a
partir de situações comuns, como um dia de chuva ou um braço quebrado, e são usados para
exemplificar os tópicos aos quais se relacionam, trazendo explicações e curiosidades para o
espectador.",
      "version": "1.0"
    },
    ...
  ]
}
```

Detalhe da aplicação (GET /rest/v1.1/app/{applicationId})

Requisição:

Parametro: tipo	**Descrição**
applicationId: long [requerido]	Identificador único da aplicação.
macAddress: string [opcional]	Endereço MAC do dispositivo, se presente e se a aplicações tiver sido avaliada através desse MAC, o campo userRating estará presente

Resposta:
200 - Ok
400 - O parâmetro `applicationId` passado é inválido
412 - O parâmetro `applicationId` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/rest/v1.1/app/5555555 HTTP/1.1
```

```
Host: gingabr.comunicacoes.gov.br
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT

{
  "app": {
    "id": 5555555,
    "name": "Vamos Fotografar! - Apolônio e Azulão",
    "promotionalText": "Ajude o Azulão a tirar fotos das plantas que dão flores e das
plantas que dão frutos! Mova a câmera com as setas do controle remoto e tire uma foto com o
botão "OK".",
    "file": 1761442,
    "fileSize": 321854,
    "rating": 3,
    "userRating": 5,
    "parentalControl": 0,
    "iconUrl" : "http://gingabr.comunicacoes.gov.br....",
    "coverUrl" : "http://gingabr.comunicacoes.gov.br....",
    "category": 3,
    "screenshots": [9, 656, 1221],
    "developerId": 123456,
    "developerName": "Televisão universitária da Unesp",
    "controlCode": 0,
    "lastChanges": "31/05/2014: Envio da aplicação.",
    "description": "Apolônio e Azulão é uma série de miniprogramas voltados ao ensino dos
mais variados tópicos ao público infantil. Os episódios tratam especialmente de curiosidades
sobre o mundo que nos cerca e sobre seu funcionamento. A série gira em torno de dois amigos:
o Professor Apolônio, um simpático e estudioso velhinho que se diverte em explicar os temas
que surgem durante suas aventuras, e Azulão, o alegre, agitado e observador ser azul que
vira-e-mexe traz curiosidades do seu variado cotidiano quando se encontra com o professor.
Os encontros desses dois personagens geram histórias que partem de momentos simples e a
partir de situações comuns, como um dia de chuva ou um braço quebrado, e são usados para
exemplificar os tópicos aos quais se relacionam, trazendo explicações e curiosidades para o
espectador.",
    "version": "1.0"
  }
}
```

Detalhe do desenvolvedor (GET /rest/v1.1/dev/{developerId})

Requisição:

Parametro: tipo	**Descrição**
developerId: long [requerido]	Identificador único do desenvolvedor.

Resposta:
200 - Ok
400 - O parâmetro `developerId` passado é inválido
412 - O parâmetro `developerId` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/rest/v1.1/dev/123456 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT

{
  "developer": {
    "id": 123456,
    "name": "Televisão universitária da Unesp",
    "email": "tvunesp@gmail.com",
    "website": "www.tv.unesp.br"
  }
}
```

Fonte do aplicativo (GET /rest/v1.1/download/app/{file})

Requisição:

Parametro: tipo	Descrição
file: long [requerido]	Identificador único do arquivo de fonte da aplicação

Resposta:
200 - Ok
400 - O parâmetro `file` passado é inválido
412 - O parâmetro `file` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/rest/v1.1/download/app/1761442 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: */*
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
Content-Disposition: attachment; filename=ApolonioEAzulao.zip
Content-Type: application/zip
Content-Length: 1556967
Date: Thu, 07 May 2015 18:47:16 GMT


[[BINARY DATA]]
```

Ícone do aplicativo (GET /thumbs/app/{applicationId})

Requisição:

Parametro: tipo	Descrição
applicationId: long [requerido]	Identificador único da aplicação.

Resposta:
200 - Ok
400 - O parâmetro `applicationId` passado é inválido
412 - O parâmetro `applicationId` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/thumbs/app/5555555 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: image/png
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: image/png
Content-Length: 36154

[[BINARY DATA]]
```

Capa do aplicativo (GET /thumbs/coverapp/{applicationId})

Requisição:

Parametro: tipo	**Descrição**
applicationId: long [requerido]	Identificador único da aplicação.

Resposta:
200 - Ok
400 - O parâmetro `applicationId` passado é inválido
412 - O parâmetro `applicationId` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/thumbs/coverapp/5555555 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: image/png
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
```

```
Content-Type: image/png
Content-Length: 36154

[[BINARY DATA]]
```

Capturas de tela do aplicativo (GET /thumbs/screenshot/{applicationId})

Requisição:

Parametro: tipo	**Descrição**
applicationId: long [requerido]	Identificador único da aplicação.
ssid: long [requerido]	Identificador unico da captura de tela.

Resposta:
200 - Ok
400 - O parâmetro `applicationId` e/ou `ssid` passados é inválido
412 - O parâmetro `applicationId` e/ou `ssid` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /portalginga/thumbs/screenshot/5555555?ssid=9 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: image/png
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: image/png
Content-Length: 36154

[[BINARY DATA]]
```

Avaliação do aplicativo (POST /rest/v1.1/rating/app/)

Requisição:

Parametro: tipo	Descrição
{ "id": 5555555, "rating": 3, "macAddress": "00:19:B9:FB:E2:57" }	Id [requerido]: Identificador único da aplicação. Rating [requerido]: Rating do usuário [0-5] macAddress [requerido]: Mac address do receptor.

Resposta:
200 - Ok
400 - O parâmetro `id` e/ou `rating` e/ou `macAddress` passados é inválido
412 - O parâmetro `id` e/ou `rating` e/ou `macAddress` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
POST /portalginga/rest/v1.1/rating/app/ HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT

{
    "id": 5555555,
    "rating": 3
}
```

Manifest da applicação (GET /rest/v1.1/manifest/{applicationId})

Requisição:

Parametro: tipo	Descrição
applicationId: long [requerido]	Identificador único da aplicação.

Resposta:
200 - Ok
400 - O parâmetro `applicationId` passado é inválido
412 - O parâmetro `applicationId` está faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /rest/v1.1/manifest/21222 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json;charset=ISO-8859-15
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT
```

[Manifest file como descrito na sessão "Estrutura do arquivo MANIFEST"]

Consultar uma atualização de aplicaçãos (GET /rest/v1.1/checkupdate/{appId}/{currentInstalledVersion})

Requisição:

Parametro: tipo	Descrição
appId: long [requerido]	Identificador único da aplicação.
currentInstalledVersion: string [requerido]	Versão atualmente instalada na caixa

Resposta:
200 - Ok
204 - Versão mais nova já instalada
400 - Os parâmetros `appId` ou `currentInstalledVersion` passados são inválido
404 - Não existe atualização para essa aplicação.
406 - Tamanho da PATCH supera o tamanho da última versão da aplicação.
412 - Os parâmetros `appId` ou `currentInstalledVersion` estão faltando

500 - Erro de servidor

Exemplo:

Requisição

```
GET /rest/v1.1/checkupdate/21222/2.4 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: */*
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/json
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT

{
    "patchSize": 13216
}
```

patchSize: Tamanho do arquivo dos arquivos da PATCH descompactado expressado em KB.

Atualizar uma aplicação através de patches (GET /rest/v1.1/update/{appId}/{currentInstalledVersion})

Requisição:

Parametro: tipo	Descrição
appId: long [requerido]	Identificador único da aplicação.
currentInstalledVersion: string [requerido]	Versão atualmente instalada na caixa

Resposta:
200 - Ok
204 - Versão mais nova já instalada
400 - Os parâmetro `appId` ou `currentInstalledVersion` passados são inválido
412 - Os parâmetro `appId` ou `currentInstalledVersion` estão faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /rest/v1.1/update/21222/2.4 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: */*
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/zip
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT
```

[Arquivo binário como descrito na seção "Estrutura padrão da patch de atualização"]

Atualizar uma aplicação por faixa através de patches (GET /rest/v1.1/update/{appId}/{minInstalledVersion}-{maxInstalledVersion})

Requisição:

Parametro: tipo	Descrição
appId: long [requerido]	Identificador único da aplicação.
minInstalledVersion: string [requerido]	Versão mínima instalada na caixa
maxInstalledVersion: string [requerido]	Versão máxima instalada na caixa

Resposta:
200 - Ok
204 - Versão mais nova já instalada
400 - Os parâmetro `appId` e/ou `minInstalledVersion` e/ou `maxInstalledVersion` passados são inválido
412 - Os parâmetro `appId` e/ou `minInstalledVersion` e/ou `maxInstalledVersion` estão faltando
500 - Erro de servidor

Exemplo:

Requisição

```
GET /rest/v1.1/update/21222/2.4-2.7 HTTP/1.1
Host: gingabr.comunicacoes.gov.br
Accept: */*
```

Resposta

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
X-Powered-By: JSF/1.2
Content-Type: application/zip
Transfer-Encoding: chunked
Date: Thu, 07 May 2015 15:28:55 GMT
```

[Arquivo binário como descrito na seção “Estrutura padrão da patch de atualização”]

Estrutura padrão da aplicação

Todas as aplicações, sejam elas embarcadas, disponibilizadas pelo portal ou transmitidas no sinal do ar devem ter a mesma estrutura a seguir:

```
|-> MANIFEST
|-> gingabr.comunicacoes.gov.br.pem
|-> gingabr.comunicacoes.gov.br.signature
|-> [portal_generico.com.br.pem]*
|-> [portal_generico.com.br.sinature]*
|-> [raiz das fontes da aplicação]
```

MANIFEST: (mandatório) é o arquivo de metadados da aplicação. Vide Estrutura do arquivo MANIFEST.

gingabr.comunicacoes.gov.br.pem: (mandatório) chave pública do portal gingabr.

gingabr.comunicacoes.gov.br.signature: (mandatório) assinatura dos arquivos contidos na aplicação. Os arquivos assinados estão especificados no campo **appSignedFiles** do arquivo **MANIFEST.**

raiz das fontes da aplicação: (mandatório) diretório contendo o código fonte da aplicaçao, nomeado como **source**.

Quando a aplicação é disponibilizada pelo portal, a estrutura acima deve estar contida em um arquivo zip.

A mesma estrutura será utilizada para envio da aplicação pelo sinal de radiodifusão. A única diferença é que, no caso do envio pelo sinal de radiodifusão, a aplicação é contida no carrossel de objetos DSMCC, em vez de um arquivo zip.

Estrutura do arquivo MANIFEST

O arquivo `MANIFEST` inclui todos os metadados que descrevem a aplicação. É utilizado JSON como sintaxe, está codificado em ISO 8859-15 (codificação utilizada no sistema SBTVD) e tem a seguinte estrutura:

```
{
     "appName" : "Nome da Aplicação",
     "appIcon" : "icon.png",
     "appDescription" : "Descrição da aplicação",
     "appType": "Ginga-NCL",
     "appEntryPoint" : "code/main.ncl",
     "appVersion" : "2.1",
     "appRating" : 0,
     "appCategory" : 0,
     "developerId" : 0000,
     "developerName" : "Nome do autor",
     "developerEmail" : "contato@app-autor.com.br",
     "developerWebSite" : "www.app-autor.com.br",
     "appSize" : 1024,
     "appSignedFiles" :  ["source/code/main.ncl", "source/code/doc.ncl",
     "source/code/script.lua",
        "source/imgs/appIco.png", "source/imgs/img01.png", "MANIFEST" ],

     "appIds" :
        [
                {
                       "host" : "gingabr.comunicacoes.gov.br",
                       "appId" : 123
                },
                {
                       "host" : "app-provider.com.br",
```

```
                    "appId" : 456
                }
        ]
}
```

A descrição dos campos do arquivo MANIFEST é a seguinte:

appName: (mandatório) string do nome da aplicação apresentável para o usuário.

appIcon: (mandatório) caminho relativo à raiz do zip que indica o arquivo de imagem para ser utilizado como ícone da applicação.

appDescription: (opcional) string contendo a descrição da aplicação.

appType: (mandatório) string indicando o tipo de aplicação (“Ginga-J” ou “Ginga-NCL”).

appEntryPoint: (mandatório) caminho do arquivo NCL inicial da aplicação NCL relativo a raiz da aplicação ou classe inicial da aplicação Java.

appVersion: (mandatório) versão da aplicação no formato <major>.<minor>.

appRating: (mandatório) Classificação indicativa do aplicativo, pode ser:
0: Livre
1: 10 anos
2: 12 anos
3: 14 anos
4: 16 anos
5: 18 anos

appCategory: (mandatório) Categoria do aplicativo, pode ser:
1: Negócios e Finanças
2: Comunicação
3: Educação
4: Entretenimento
5: Corpo e Saúde
6: Mídia
7: Social

developerId: (opcional) Identificador da pessoa ou entidade responsável pelo desenvolvimento da aplicação.

developerName: (opcional) Nome da pessoa ou entidade responsável pelo desenvolvimento da aplicação.

developerEmail: (opcional) e-mail de contato da pessoa ou entidade responsável pelo desenvolvimento da aplicação.

developerWebSite: (opcional) website da pessoa ou entidade responsável pelo desenvolvimento da aplicação.

appSize: (mandatório) valor numérico, tamanho da aplicação descompactada expressado em KB.

appSigneFiles: (mandatório) lista de arquivos da aplicação assinados (a ordem é relevante, para gerar a assinatura e receptor fazer a validação) relativos a raiz do zip. O MANIFEST sempre deve ser incluso na assinatura.

appIds: (mandatório, é requerido no mínimo um identificador) vetor de identificadores da aplicação. Pelo menos um elemento deve ser definido.

appIds[i].host: (mandatório) URL do portal que disponibiliza a aplicação.

appIds[i].appId: (mandatório) identificador da aplicação atribuída pelo portal. Cada aplicação deve ter um id único no escopo de um portal.

Estrutura padrão da patch de atualização

Uma patch de atualização é um arquivo de diferenças destinados a atualizações de aplicativos.

```
|-> PATCH
|-> MANIFEST
|-> gingabr.comunicacoes.gov.br.pem
|-> gingabr.comunicacoes.gov.br.signature
|-> gingabr.comunicacoes.gov.br.patch.signature
|-> [portal_generico.com.br.pem]*
|-> [portal_generico.com.br.sinature]*
```

```
|-> [portal_generico.com.br.patch.sinature]*
|-> [raiz dos arquivos de diferença]
```

PATCH: (mandatório) é o arquivo de metadados da patch. Vide Estrutura do arquivo PATCH.

MANIFEST: (mandatório) é o arquivo de metadados da aplicação resultante (após aplicação da patch). Vide Estrutura do arquivo MANIFEST.

gingabr.comunicacoes.gov.br.pem: (mandatório) chave pública do portal gingabr.

gingabr.comunicacoes.gov.br.signature: (mandatório) assinatura dos arquivos contidos na aplicação resultante. Os arquivos assinados estão especificados no campo **appSignedFiles** do arquivo **MANIFEST.**

gingabr.comunicacoes.gov.br.patch.signature: (mandatório) assinatura dos arquivos contidos na patch. Os arquivos assinados estão especificados no campo **patchSignedFiles** do arquivo **PATCH.**

raiz dos arquivos de diferença: (mandatório) diretório contendo os arquivos de diferença, nomeado como patch.

Quando a patch é disponibilizada pelo portal, a estrutura acima deve estar contida em um arquivo zip.

A única diferença do envio da aplicação pelo sinal de radiodifusão é que a aplicação é contida no carrossel de objetos DSMCC, ao invés de um arquivo zip.

Estrutura do arquivo PATCH

O arquivo PATCH inclui as diferenças entre duas versões de uma aplicação

```
{
    “appVersion” : “3.1”,
    “appVersionFrom” : “2.0”,
    “minInstalledVersion” : “1.2”,
    “maxInstalledVersion” : “2.0”,
    “appSize” : 1024,
    “patchSize” : 5,
```

```
"host": "gingabr.comunicacoes.gov.br"
"patchSignedFiles": ["patch/code/main.ncl", "patch/code/doc.ncl",
 "patch/code/script.lua",
    "patch/imgs/appIco.png", "patch/imgs/img01.png", "PATCH" ],
"diffs": {
    "1.2": {
        "test": [
           {
               "path": "code/main.ncl",
               "digest":
"bf30595fdb0f9d4a283e5ff5668dbc4b221f8f29909b0905b1df7aaaa89e0f9c"
           }
        ],
        "remove": ["imgs/appIco.png", "imgs/img01.png"],
        "add": ["code/main.ncl", "code/doc.ncl", "code/script.lua"]
    },
    "1.6": {
        "test": [
           {
               "path": "code/main.ncl",
               "digest":
"bf30595fdb0f9d4a283e5ff5668dbc4b221f8f29909b0905b1df7aaaa89e0f9c"
           }
        ],
        "remove": ["imgs/appIco.png", "imgs/img01.png"],
        "add": ["code/main.ncl", "code/doc.ncl", "code/script.lua"]
    },
    "2.0": {
        "test": [
           {
               "path": "code/main.ncl",
               "digest":
"bf30595fdb0f9d4a283e5ff5668dbc4b221f8f29909b0905b1df7aaaa89e0f9c"
           }
```

```
            ],
            "remove": ["imgs/appIco.png", "imgs/img01.png"],
            "add": ["code/main.ncl", "code/doc.ncl", "code/script.lua"]
        },
    },
    "appName" : "Novo Nome da Aplicação",
    "appIcon" : "icon.png",
    "appDescription" : "Nova descrição da aplicação",
    "appType": "Ginga-NCL",
    "appEntryPoint" : "code/new-main.ncl",
    "appRating" : 0,
    "appCategory" : 0,
    "appIds" : [
        {
            "host" : "gingabr.comunicacoes.gov.br",
            "appId" : 123
        },
        {
            "host" : "app-provider.com.br",
            "appId" : 456
        }
    ]
}
```

A descrição dos campos do PATCH é a seguinte:

appIds: (mandatório) os ids da aplicações em diversos hosts

appVersion: (mandatório) a versão da aplicação depois que a atualização estiver concluída

appVersionFrom: (opcional) a versão da aplicação a partir da qual a atualização deve ser feita. Campo mandatório para atualizações de patch

minInstalledVersion: (opcional) versão mínima da aplicação a partir da qual a atualização deve ser feita. Campo mandatório para atualizações de patch por faixa

maxInstalledVersion: (opcional) versão máxima da aplicação a partir da qual a atualização deve ser feita. Campo mandatório para atualizações de patch por faixa

appSize: (mandatório) novo tamanho da aplicação depois de atualizada. Expressado em KB.

patchSize: (mandatório) tamanho da patch a ser baixada

patchSignedFiles: (mandatório) lista de arquivos da aplicação assinados (a ordem é relevante, para gerar a assinatura e receptor fazer a validação) relativo a raiz do zip. O arquivo PATCH sempre deve ser incluso na assinatura.

host: (opcional) host provedor da patch, caso não exista será considerado o domínio do qual a patch foi baixada.

diffs: (mandatório) objeto de diferenças entre uma versão e outra que devem ser executado na seguinte sequencia: `tests`, `removes` e `adds`. Todos os caminhos desse campo são relativos à raiz do sistema de arquivos da aplicação.

Existe um objeto com `tests`, `removes` e `adds` para cada versão que esse patch se aplica. As versões estão contidas tanto na faixa de versões delimitada pelo `minInstalledVersion` e `maxInstalledVersion` quanto na definida pelo `appVersionFrom`.

O atributo **test** é um array com de testes a serem realizados antes de qualquer alteração seja realizada. Cada teste é um objeto com dois atributos `path` e `digest`. Os testes só são realizados em arquivos locais.

```
{
    “path”: “code/main.ncl”,
    “digest”:
“bf30595fdb0f9d4a283e5ff5668dbc4b221f8f29909b0905b1df7aaaa89e0f9c”
}
```

O atributo **path** traz o caminho do arquivo a ser testado.

O campo **digest** contem o checksum do arquivo a ser testado. O checksum é calculado utilizando o algoritimo sha256.

O atributo **remove** contem um array de caminhos para arquivos locais que devem ser removidos.

O atributo **add** contem um array de caminhos para arquivos da patch a serem adicionados na atualização.

appName: (circunstancial) string do nome da aplicação apresentável para o usuário. Só deve existir se o nome da aplicação mudar

appIcon: (circunstancial) caminho relativo à raiz do zip que indica o arquivo de imagem para ser utilizado como ícone da applicação. Só deve existir se o caminho do ícone da aplicação mudar. Se o arquivo for alterado mas o caminho for o mesmo, não deve existir.

appDescription: (circunstancial) string contendo a descrição da aplicação. Só deve existir se a descrição da aplicação mudar

appType: (circunstancial) pode ser Ginga-J ou Ginga-NCL. Só deve existir se o tipo da aplicação mudar.

appEntryPoint: (circunstancial) caminho ao arquivo NCL inicial da aplicação NCL ou classe inicial da aplicação Java. Só deve existir se entryPoint mudar entre uma versão e outra

appRating: (circunstancial) Classificação indicativa do aplicativo, pode ser:
0: Livre
1: 10 anos
2: 12 anos
3: 14 anos
4: 16 anos
5: 18 anos

Só deve existir se a classificação indicativa mudar entre as duas versões

appCategory: (circunstancial) Categoria do aplicativo, pode ser:
1: Negócios e Finanças
2: Comunicação
3: Educação
4: Entretenimento
5: Corpo e Saúde
6: Mídia
7: Social

Só deve existir se a categoria da aplicação mudar

Atribuição e gerenciamento de identificadores de aplicação

O Portal Ginga será responsável por atribuir um identificador numérico único (a unicidade deve ser garantida dentro do escopo do portal) para cada aplicação que irá distribuir.
Da mesma forma, outros portais que venham distribuir as mesmas ou outras aplicações, deveram atribuir um identificador numérico único no escopo de cada portal.
O identificador final da aplicação é composto pela combinação do identificador numérico atribuído pelo portal e o nome de domínio do host, no formato: `<appId>@<dominio_do_portal>`. Os exemplos a seguir, são de identificadores de aplicação diferentes:

```
123@ebc.com.br
123@gingabr.comunicacoes.gov.br
```

No arquivo `MANIFEST` podem ser declarados mais de um identificador de aplicação (no campo `appIds`), caso a aplicação venha a ser distribuída por diferentes portais. Neste caso, a mesma aplicação terá múltiplos identificadores.

Segurança e assinatura das aplicações

Cada portal (ou por cada `appId`) declarado no `MANIFEST` deve prover uma certificado digital e uma assinatura para todos os arquivos da aplicação, inclusive o manifest.

É requerido que todos os certificados utilizados para gerar as assinaturas tenham como raiz o ICP Brasil e devem ser outorgados para o nome domínio de cada portal.

Todas as aplicações enviadas ao Portal Ginga serão assinadas com um certificado emitido para o host `gingabr.comunicacoes.gov.br`. Outras assinaturas podem também ser utilizadas, caso as aplicações sejam disponibilizadas por outros portais.

Cálculo da Assinatura

As assinaturas devem ser calculadas na ordem em que os arquivos são declarados no campo `appSignedFiles`, utilizando o certificado de cada portal. O hash resultante deve ser colocado

em arquivo na raiz do sistema de arquivos da aplicação e nomeado da seguinte forma `<dominio_do_portal>.signature`.

O arquivo `MANIFEST` deve ser sempre incluído no cálculo da assinatura e é requerido que seja listado no campo `appSignedFiles`, do próprio arquivo `MANIFEST`.

Para o cálculo da assinatura, o algoritmo utilizado deve ser o `SHA256`. É feito o `checksum` de cada arquivo declarado no campo `appSignedFiles` e o seu restulado é colocado em um arquivo temporário separado por `\n`. Após isso, esse arquivo temporário é assinado com o certificado do portal.

O mesmo processo utilizado para calcular os arquivos `*.signature` também é utilizado para gerar os arquivos `*.patch.signature`, porém usa-se o campo `patchSignedFiles` do arquivo `PATCH`.

A assinatura nos arquivos `*.signature` deve estar no formato OpenPGP definido em RFC4880.

Exemplo de arquivo `*.signature`:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAABAgAGBQJVS85jAAoJEE4UOenTlKl0WJcIAIyq5NYfBrVlDT1v3Vz/8kg2
mBHeYgPhCPR7NcWLjSVt6VcIjcfEGUCP8ZpR0aWyAPV+pW1caJme2gTWE9CyDmR9
jWBTDsLnfQ1AI5HanEz5JHvCS1RSs+EgAc3i9ZFSNhXGazNs/IVWMaM6YFaRWyYX
tVx5TwM43LfdmoEASyGvbQVLceteut+B+5Yw50X4V7hT4I8QiM22uKbHtR6K5kQ9
rnLGwmEL4iTDxq955CrxmcAIr24fjTPdcgABK1DQMqc26UCwsgutKkcBPSUwp+gj
q0tfqwR5crFuvA0XzyCzummA2uApRYrjuGe8T7EvqNfmhK2VEm8FI9QuKLR7vhQ=
=VjG7
-----END PGP SIGNATURE-----
```

Os set-top-boxes distribuídos devem ter capacidade para checar a autenticidade de certificados e assinaturas gerados com certificados derivados do ICP Brasil.

Atualização das aplicações

Vinculação da aplicação no receptor

Quando as aplicações são descarregadas no receptor, independente do meio ou origem (Portal Ginga, outros portais, sinal de TV Digital, embarcada pelo fabricante, etc), o receptor deverá procurar uma correspondência de pelo menos um dos `appIds` listados no `MANIFEST` com os `appIds` dos `MANIFEST` declarados nas aplicações previamente instaladas no receptor.

Existindo pelo menos uma correspondência, então trata-se da mesma aplicação.

Nota: as assinaturas podem ser diferentes, já que podem ser versões diferentes da mesma aplicação.

Atualização das aplicações via Internet (Portal Ginga ou outros portais)

Aplicação inteira

O processo de atualização de aplicação compreende os seguintes passos:

1. O receptor descarrega o `MANIFEST` da aplicação a atualizar, e verifica que a versão é maior da declarada no `MANIFEST` da aplicação instalada.
2. O receptor descarrega o zip da aplicação completa.
3. O receptor valida todas as assinaturas da aplicação descarregada.
4. Só depois de verificar **todas** as assinaturas com sucesso, o receptor substitui os arquivos da aplicação instalada pelos arquivos da aplicação descarregada.

Patch

O processo de atualização de aplicação através de patch compreende os seguintes passos:

1. Verificar a autenticidade/integridade do arquivo `PATCH` através do arquivo de assinatura utilizando a chave pública;
2. Verificar se a versão atualmente instalada é a mesma contida no atributo `appVersionFrom` ou se está dentro da faixa de atualização dos atributos `minInstalledVersion` e `maxInstalledVersion` do arquivo `PATCH`. Em caso negativo, deve-se interromper a atualização;
3. Realizar os testes dos arquivos da aplicação antiga utilizando os dados do atributo **test** do objeto **diffs**, da versão alvo, do arquivo `PATCH`. Caso algum teste falhe, deve-se abortar o processo de atualização. Cada teste consiste em uma verificação de um

checksum `SHA256` dos arquivos descrito pelos atributos path o resultado do checksum está no atributo digest;

4. Remover todos os arquivos locais que estão relacionados no array **remove** do arquivo `PATCH`;
5. Mover os arquivos da patch relacionados no atributo **add** do arquivo `PATCH` para a instalação local;
6. Copiar o novo arquivo de `MANIFEST` para o local da aplicação;
7. Após a realização das operações do atributo **diff**, realizar o checksum da aplicação resultante como um todo, utilizando os arquivos `[portal_generico.com.br.pem]` e`[portal_generico.com.br.sinature]`, com esta verificação será assegurada a integridade da aplicação após a atualização.

Atualização das aplicações via sinal de TV Digital

Aplicação inteira

O processo de atualização de aplicação transmitida pelo sinal de TV Digital compreende os seguintes passos:

1. A aplicação é transmitida conforme as normas ABNT para transmissão de aplicações Ginga:
   i. Caso não seja desejável que a aplicação seja executada, pode ser utilizado um dos códigos de controle PRESENT, PREFETCH, STORED ou STORED_PRESENT.

   ii. Caso a aplicação não deva ser listada na lista de aplicações, pode ser configurado o campo `visibility_flag` da tabela AIT para um dos valores `00` ou `01`.

   iii. Também, caso seja desejável um comportamento diferenciado da aplicação em receptores legado, ou quando a aplicação não está instalada, um ponto de entrada (campo `initialClassByte`) diferente pode ser configurado na tabela AIT.

   iv. O campo `application_name` da tabela AIT deve ser definido da seguinte forma: `<nome_legível>(<appId>_<appVersion>@<host>)`. Onde:
      iv.a) **nome_legível:** Nome da aplicação;
      iv.b) **appId:** identificador da aplicação atribuído pelo portal;
      iv.c) **appVersion:** versão da aplicação;
      iv.d) **host:** nome de domínio da do portal responsável pela aplicação.

A modo de exemplo, a aplicação Brasil 4D, com versão 2.5, identificada pelo id 123, no portal gingabr.comunicacoes.gov.br, deve ser nomeada da seguinte forma na transmissão via AIT: “Brasil 4D (123_2.5@gingabr.comunicacoes.gov.br)”

2. O receptor descarrega o carrossel de objetos DSMCC da aplicação.

3. Caso o receptor encontre um arquivo `MANIFEST` na raiz do carrossel de objetos DSMCC, tentará fazer uma correspondência com alguma das aplicações previamente instaladas.

4. Caso consiga uma correspondência, verificará que a versão é maior da declarada no `MANIFEST` da aplicação instalada.

5. Caso a versão seja maior, o receptor valida todas as assinaturas da aplicação descarregada.

6. Só se todas as assinaturas são válidas, o receptor substitui os arquivos da aplicação instalada pelos arquivos da aplicação descarregada pelo sinal de TV Digital.

Patch

Sinalização de atualização de aplicação instalada

Deve ser definido um Maker_id comum para todos os receptores que implementem a funcionalidade de atualização de aplicações instaladas. Para isto, o valor de maker_id deve ser reservado na Tabela 77 da Norma ABNT NBR 15608-3.

O valor a utilizar para o Maker_id:

Nome do Fabricante	**Valor maker_id**
MINC	0x06 (00000110b)

Da mesma forma, deverá ser utilizado um model_id comum com o valor de 0x01.

A sinalização é feita mediante a tabela SDTT (Software Download Trigger Table), conforme definida na Norma ABNT NBR 15608-3, seção 29.4. A SDTT a ser utilizada será a definida para a camada de alta proteção, destinada a notificação de atualizações. A estrutura da tabela SDTT é definida pela Tabela 60 da Norma ABNT NBR 15608-3. Os valores a serem atribuídos a cada um dos campos, assim como configurações de transmissão gerais são definidos a seguir:

Campo	Valor
PID	0x0028
table_id	0xC3
section_syntax_indicator	1b
reserved_for_future_use	1b
reserved	11b
section_length	(de acordo com o tamanho da seção)
table_id_extension (maker_id model_id)	0x0601
reserved	11b
version_number	(de acordo com a versão da seção)
current_next_indicator	1b
section_number	(de acordo com o número de seções transmitidos)
last_section_number	(de acordo com o número de seções transmitidos)
transport_stream_id	mesmo valor que o transmitido na PAT
original_network_id	mesmo valor que o transmitido na NIT
service_id	Identificador do serviço no qual a atualização é transmitida
num_of_contents	número de atualizações transmitidas no serviço
group	0001b (não utilizado)
target_version	0x00 (não utilizado)
new_version	versão da atualização
download_level	01b: atualização obrigatória 00b: atualização opcional
version_indicator	00b (não utilizado)
content_descriptor_length	(de acordo com o tamanho do download_content_descriptor())
schedule_descriptor_length	0x00
schedule_time-shift_information	0xF
descriptor()	Deve conter uma ocorrência de download_content_descriptor()

Tamanho máximo da seção: 4 KBytes

Taxa máxima de transmissão: 2 Kbps

Na SDTT podem ser sinalizadas múltiplas atualizações no loop de conteúdos desta tabela.

O download_content_descriptor(), definido na Tabela 64 da ABNT NBR 15608-3 deve ser parametrizado da seguinte forma:

download_content_descriptor {	
descriptor_tag	0xC9
descriptor_length	de acordo com o tamanho do descritor
reboot	0b
add_on	0b
compatibility_flag	0b
module_info_flag	0b
text_info_flag	0b
reserved	111b
component_size	Tamanho do carrossel de objetos transmitindo a atualização
download_id	Download ID do carrossel de objetos transmitindo a atualização
time_out_value_DII	0x0000 (não utilizado)
leak_rate	0x0000 (não utilizado)
reserved	11b
component_tag	Component TAG, como definida na PMT do serviço, que identifica o fluxo elementar do carrossel e objetos transmitindo a atualização
private_data_length	Tamanho da estrutura AppPortalUpdate
private_data	Deve ser preenchido com a estrutura AppPortalUpdate()
}	

A estrutura AppPortalUpdate(), que deve ser transmitida no campo private_data do download_content_descriptor() , é definida a continuação:

AppPortalUpdate() {	No. de bits
app_IDs_length	8
for (i=0; i<N; i++) {	
app_IDs() {	
host_length	8
for (i=0; i<N; i++) {	
host_char	8
}	
app_ID	64
}	
}	
app_version() {	

app_major_version	8
app_minor_version	8
}	
app_size	32
app_patch_size	32
reserved	7
app_min_installed_version_flag	1
if (app_min_installed_version_flag == 1) {	
app_min_installed_version() {	
app_min_installed_major_version	8
app_min_installed_minor_version	8
}	
}	
reserved	7
app_max_installed_version_flag	1
if (app_max_installed_version_flag == 1) {	
app_max_installed_version() {	
app_max_installed_major_version	8
app_max_installed_minor_version	8
}	
}	

app_IDs_length: campo de 8 bits que deve obrigatoriamente indicar o comprimento em bytes da lista de IDs da aplicação em diversos hosts.

app_IDs(): Vetor de identificadores da aplicação. Pelo menos um elemento deve ser definido.

host_length: campo de 8 bits que deve obrigatoriamente indicar o comprimento em bytes do nome do host (URL do portal que disponibiliza a aplicação)

host_char: campo de 8 bits que especifica o texto de descrição da URL do portal que disponibiliza a aplicação

app_ID: campo de 64 bits com o identificador da aplicação atribuída pelo portal. Cada aplicação deve ter um id único no escopo de um portal.

app_version(): versão da aplicação no formato <major>.<minor>.

app_major_version: campo de 8 bits que indica o número principal de versão da aplicação.

app_minor_version: campo de 8 bits que indica o número secundário de versão da aplicação.

app_size: campo de 32 bits que indica o novo tamanho da aplicação depois de atualizada. Expressado em KB.

app_patch_size: campo de 32 bits que indica o tamanho do patch a ser baixado. Expressado em KB.

app_min_installed_version_flag: quando este campo de 1 bit for igual a “1” o application_min_installed_version deve obrigatoriamente ser definido.

app_min_installed_version(): (opcional) a versão da aplicação a partir da qual a atualização deve ser aplicada. Deve ser o menor valor de minInstalledVersion dentre os arquivos de patch transmitidos no carrossel de objetos. Obrigatório na transmissão de patches. Formato <major>.<minor>.

app_min_installed_major_version: campo de 8 bits que indica o número principal de versão da aplicação mínima no qual o patch pode ser aplicado.

app_min_installed_minor_version: campo de 8 bits que indica o número secundário de versão da aplicação mínima no qual o patch pode ser aplicado.

app_max_installed_version_flag: quando este campo de 1 bit for igual a “1” o application_max_installed_version deve obrigatoriamente ser definido.

app_max_installed_version(): (opcional) a versão máxima da aplicação que a atualização pode ser aplicada. Deve ser o maior valor de maxInstalledVersion dentre os arquivos de patch transmitidos no carrossel de objetos.Obrigatório na transmissão de patches. Formato <major>.<minor>.

app_max_installed_major_version: campo de 8 bits que indica o número principal de versão da aplicação máximo no qual o patch pode ser aplicado.

app_max_installed_minor_version: campo de 8 bits que indica o número secundário de versão da aplicação máximo no qual o patch pode ser aplicado.