ABNT: ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS
CB: 08 COMITÊ BRASILEIRO DE AERONÁUTICA E ESPAÇO
SC: 08.001 SUBCOMITÊ DE ATIVIDADE ESPACIAL
CE: 08:010.70 COMISSÃO DE ESTUDO EM SISTEMAS ESPACIAIS DE TRANSFERÊNCIA DE DADOS E DE INFORMAÇÃO

## Space Data and Information Transfer Systems

# System for IBI Generation

**Commission Technical Reports – 10-I (RTC-10-I)**
**– Editorial Version – 1: July 31, 2015 –**
**(English translation of RTC-10-a)**

**CE 08:010.70 Commission-of-Study in**
**Space Data and Information Transfer Systems**

**SPACE DATA AND INFORMATION TRANSFER SYSTEMS**
**SYSTEM FOR IBI RESOLUTION**

# CONTRIBUTORS OF THIS ISSUE

The publishing of this internal document, called Commission Technical Reports N° 10-I, or RTC-10-I, had the participation of the following MEMBERS of Commission-of-Study in Space Data and Information Transfer Systems – ABNT CE 08: 010.70:

| | | |
|---|---|---|
| Cíntia Borges Margi | Member | LARC/EPUSP |
| Danilo C. Carvalho | Member | ANATEL |
| Eduardo W. Bergamini | Coordinator | INPE/MCTI |
| Gerald J. F. Banon | Member | INPE/MCTI |
| João Manoel R. Zaninotto | Member | EMBRAER |
| Leandro Vaz Barros Reis | Member | ANATEL |
| Marco Antonio Grivet M. Maia | Member | PUC-RIO |
| Marília Vidigal da Costa Souza | Member | EMBRAER |
| Mauricio G. Vieira Ferreira | Member | INPE/MCTI |
| Regina M. Silveira | Observer | LARC/EPUSP |
| Reginaldo Palazzo Júnior | Member | UNICAMP |
| Sérgio Costa | Member | IAE/DCTA |
| Valéria Cristina M. N. Leite | Member | IAE/DCTA |
| Rogério Pirk | Member | IAE/DCTA |
| Lília de Sá Silva | Member | INPE |
| Sergio Fugivara | Member | IAE/DCTA |
| Walter Abrahão dos Santos | Member | INPE |
| Roberto Roma de Vasconcellos | Member | IAE/DCTA |
| Mateus Mosca Viana | Member | UNIFOR |
| Marcello Silva de Abreu | Member | EMBRAER |
| Antônio Cassiano Filho | Member | INPE |

The Secretary of RME/TEC organ of INPE, who runs the secretariat of this ABNT Commission-of-Study, conducted na extensive work of publishing and correspondence support, in preparing this document:

| | | |
|---|---|---|
| Helen Joyce Aparecida | Secretary | RME/TEC/INPE |

São José dos Campos, May, 2015.

## AUTHOR'S NOTE

This document reflects the result of a long maturation process of a definition of an identification system of information items.

The original idea, conceived in 1995, seeks to harness the own existing infrastructure offered by Internet.

Since its conception, the realization of this idea was being gradually improved. This whole process has resulted in this proposal, which has been largely tested in a computational platform called UR*Lib*. The UR*Lib* platform is the one adopted by INPE to host its Scientific Memory.

It is worth mentioning that although this idea has been primarily designed to identify information items, in the case, under a standard character, the identification system associated with it, as defined in this document, could also be used to identify any item or object, so, also in a standardized way, in principle.

The original document that gave rise to this standard that is being recommended has the title: "Identificador com base na Internet (IBI): Sistema de identificação" (Internet based Identifier (IBI): Identification system). It was published by INPE in the form of research report, under the identification code: iconet.com.br/banon/2009/09.09.22.01-RPQ. It is accessible from pointing at the URL: <http://urlib.net/LK47B6W/362SFKH>. The present document is the traduction of the technical report RTC-10-a (2nd edition) entitled: "Sistema para geração de IBI", which is accessible from pointing at the URL: <http://urlib.net/8JMKD3MGP8W/3JUK862>.

## ABSTRACT

This standard presents a procedure that leads to the creation of two versions of a global identifier, which is intended, in a long term, to consistently and compactly identify and to provide a convenient access to various kinds of information items (documents, maps, images, etc.), which are typically stored in collections, as found in digital repositories, in archives, or elsewhere. The practical deployment of this global identifier conveniently and essentially solely requires, at no additional cost, the widely, already available infrastructure of the Internet. This global identifier can be used in combination with information storage systems, which deal with collections and which, in this way, may enable remarkable simplicity in the processes dedicated to the creation of copies in different collections, also including simplicity in the migration of information items among such collections. In particular, a variety of convenient applications of a global identifier of this nature in space data and information systems are envisioned.

## LIST OF FIGURES

**Page**

## LIST OF TABLES

**Page**

## CONTENTS

## 1 Introduction

This standard presents two forms in which a global identifier can be created to identify and provide consistent and enduring access to various types of information items (documents, maps, images, etc.) stored in collections like those found in digital repositories, archives, or other information entities.

The implementation of this global identifier requires, in a roundabout way, the existing and readily available Internet infrastructure. Therefore, without additional cost, in this regard.

This global identifier can be used in association with the information storage process in collections. What also makes it simple to create copies in separate collections, including the information item migration itself between such collections.

The various applications of a global identifier of this nature are also of particular interest for use in spatial data and information systems.

## 2 Scope

This standard describes and allows to create an identification system based on the Internet that associates to each item of information to be identified, a label that can be used as an identifier for that item. The rules for the construction of two presentation forms of the mentioned label are also presented herein.

## 3 Rationale

It is assumed that hyperlinks or simply links or pointers, essential elements in the navigation between information items (documents, maps, images, etc.) currently available on the Internet should have its operation preserved and available for long term.

The solution in order to guarantee long life in preserving pointers, so with persistence and durability, implies the use of a global identification system.

The physical addressing system for an information item on the web by means of a URL (Uniform Resource Locator) is not a persistent identification system, because with time, the location of an information item can change, hence the association: "information item" $\mapsto$ URL, do not have a long-lasting character, so that it may not be essentially permanent.

Once an identification system has been chosen, in such a way that labels can be assigned to information items, the issue of the construction of persistent pointers might be solved using a resolution system, which must have the basic purpose of redirecting each URL, now containing just the identifier of an information item, to the URL which effectively contains its physical address.

## 4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

**IBI**: acronym for "Internet Based Identifier". Any **label** generated by the **system for IBI generation**.

**IBI of an item**: **label** assigned to an **item** by the **system for IBI generation** using the addressing by domain name or IP.

**IBIp of an item**: **label** assigned to an **item** by the **system for IBI generation** using addressing by IP.

**Identifier of an item**: **label** assigned to an **item** by a **identification system**.

**Identification subsystem**: any **identification system** restricted to a subset of **items**.

**Identification system**: any injective function from a set of **items** and a set of **labels**, associating to each **item** the **identifier of that item**.

**Identification system on two levels**: any **identification system** associating an **item** to a **label** obtained from a pair of **labels**, the first identifying the **identification subsystem** responsible for identifying the **item**, and the second being the **label** assigned to the **item** by this **identification subsystem**.

**Installer**: function from a set of **identification subsystems** and a set of string pairs telling the computer name or IP, and the access door, where the **label generator** used by a particular **identification subsystem** was installed.

**Information item**: any **item** consisting exclusively of digital data, i.e., any digital data to be identified. For example, documents, maps, images, etc. in digital format.

**Item**: any object to be identified.

**Label**: any finite string chosen within a finite alphabet, used as **identifier of an item**.

**Label generator**: injective function used by an **identification system** to generate the **identifier of an item**.

**Name of the uniform repository of an item**: **identifier of an item** that can

be used to store it digitally in a file system, whenever it is an **information item**. **Label** assigned to an **item** by **system for IBI generation** using the addressing by domain name.

**Spatial distributor**: function between a set of **items** and a set of **identification subsystems**, distributing each **item** to a given **identification subsystem**, making this **identification subsystem** responsible for identifying this **item**.

**System for IBI generation**: any of the two **identifications system on two levels** object of this standard.

**Temporal distributor**: function from a set of **items** and a set of dates expressed in a fraction of a second, distributing each **item** along a timeline.

## 5 Description of the system for IBI generation

In this standard, the **items** (objects to be identified) are considered forming sets. For example, a set of folders.

In turn, the **labels** used to identify the **items** are considered forming finite or countable sets. For example, all the strings of a maximum of 255 alphanumeric characters, or integers of the set representing dates expressed in seconds.

For being an injective function, an **identification system** associates, on a permanent basis, each **item** to a single **label**, so that, separate **items** are associated to separate **labels**.

By the restriction that the sets of **labels** are finite (respectively, countable), and by the property that an **identification system** is injective, the **item** set must necessarily be finite (respectively, countable).

The **identification system**, object of this report, consists of four main components: a set of **identification subsystems**, a **spatial distributor** of **items**, an **identification system** of **identification subsystems** and a **label generator**. Together they form an **identification system on two levels** described in detail below.

Let $I$ be the set of **items** to be identified.

Let $S$ be the set of **identification subsystems**.

Let $R_1$ be a finite set of **labels**.

Let $R_2$ be a countable set of **labels**.

Let $R$ be a countable set of **labels**.

Let $f : S \to R_1$ be the **identification system** of the **identification subsystems**.

Let $g : I \to S$ be the **spatial distributor** defining which **identification subsystem** is responsible for the identification of each **item**. Hence, for any $s \in S$, the set of **items** under the responsability of the **identification subsystem** $s$ is $g^*(s)$, the fiber of $s$ under $g$ (see the fiber definition in Appendix A).

Let, for any $i \in I$, $g(i) : g^*(g(i)) \to R_2$, be the **identification subsystem** responsible for the identification of the **item** $i$, within the scope of this subsystem.

Let $h : I \to R_1 \times R_2$ be the function defined by:

$$h(i) \triangleq (f(g(i)), g(i)(i)), \qquad \text{for any } i \in I.$$

Let $c : R_1 \times R_2 \to R$ be the **label generator** concatenating reversibly (i.e., $c$ is injective) the labels from $R_1$ and $R_2$ .

An **identification system on two levels** is the functin $s : I \to R$ defined as the composition of $h$ and $c$, i.e., by: $s \triangleq c \circ h$.

Given an **item** $i$ in $I$, an **identification system on two levels** assigns to $i$ the **label** generated by $c$ from the pair $(f(g(i)), g(i)(i))$ consisting, on one hand, by the **label** $f(g(i))$ assigned by the **identification system** $f$ to the **identification subsytem** $g(i)$ (reponsible by the identification of $i$ within the scope $g^*(g(i))$), and, on the other hand, by the **label** $g(i)(i)$ assigned by the **identification subsystem** $g(i)$ to the **item** $i$.

The two types of **system for IBI generation**, object of this standard, are particular cases of a **identification system on two levels** whose components are shown in Figure 5.1.

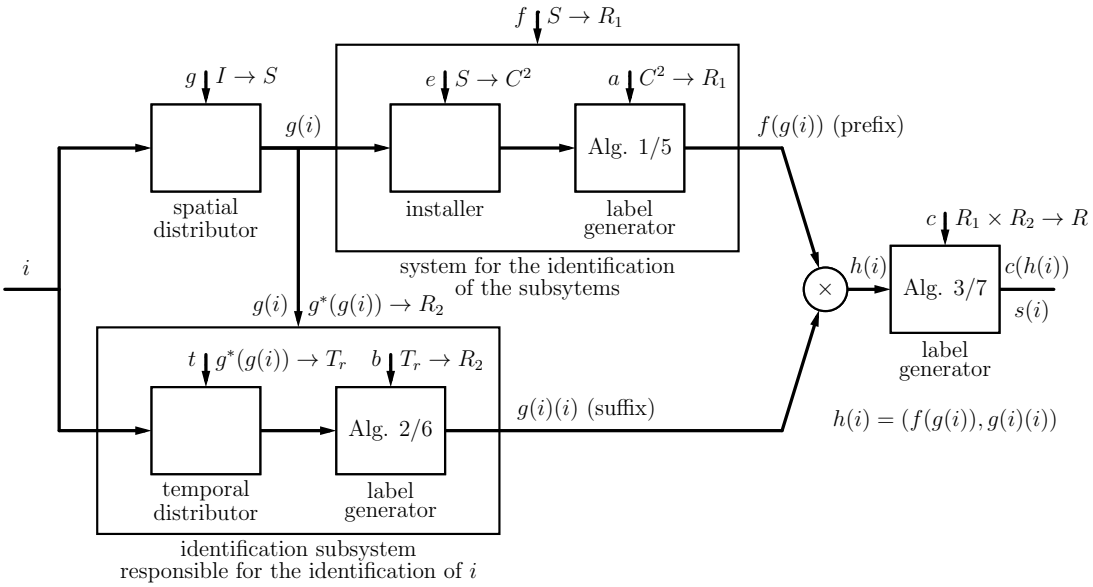


Figure 5.1 - System for IBI generation

In Figure 5.1, each box represents a function, for example $g$ (indicated by a small

arrow above the box), with its input (on the left side), for example $i$, and its output (on the right side), for example $g(i)$.

While $g(i)(i)$, the **identifier of an item** $i$, provided by the **identification subsystem** $g(i)$, is valid only within the scope $g^*(g(i))$ of this subsystem, $s(i)$, the **identifier of this item**, provided by the **identification system on two levels** $s$, is valid within the overall scope $I$.

The **identifier of an item** will be obtained as a result of a request to a server responsible for a certain **identification subsystem**, hosted exclusively on a computer having a fully qualified domain names. In this context, each of the **identification subsystems** is identified globally by a domain name or an IP (Internet Protocol) on the Internet (and a port), allowing to build in this way, simply, the prefix.

The suffix, provided by an **identification subsystem**, follows a common rule to all subsystems, and is built based on the date and time of the association of the **item** to the **label**.

The **label** $s(i)$ assigned to the **item** $i$ by the **identification system on two levels** $s$ is called, in this standard, of "Internet Based Identifier" or **IBI**, and $s(i)$ is the **IBI of the item** $i$.

In this standard, two types of prefix inherited from Internet are considered.

The first type consist of building the **identifier of an identification subsystem**, i.e., the prefix, based on the computer's domain name[1] hosting the server responsible for this subsystem, as well as the port of access to that server.

In the second type, the prefix is obtained based on the computer's IP, instead of the domain name.

The following real examples anticipate some of the details on the identifier formation that will be given in the next two chapters.

**Example 1** (identifier based on the domain name) − The association of an **item** to a suffix, which occurred on February 16, 2009 at 17 hours 46 minutes[2] resulted in the suffix:

[1]The domain name may eventually refer to the domain name of a virtual computer (virtual host).

[2]Date and time expressed in Coordinated Universal Time (UTC).

2009/02.16.17.46

The server issuing this suffix was hosted on a computer with domain name mtc-m18.sid.inpe.br, and accessible from the port 80, leading to the use of the prefix:

sid.inpe.br/mtc-m18@80

In this way, the **identifier for the item** became:

sid.inpe.br/mtc-m18@80/2009/02.16.17.46

□

It can be observed that even if the domain name mtc-m18.sid.inpe.br of Example 1 turns to be abandoned or changes hands, this does not preclude the use of the **identifier created for that item**. The important thing is that these data were relevant in the context of the Internet on the date and time of the association between the **item** and its **label**. The same also applies to the second following example that illustrates the formation of an identifier based on the IP.

**Example 2** (identifier based on the IP) − The association of an **item** to a suffix, which occurred on the second 1234806360 in POSIX time (corresponding to date of February 16, 2009 at 17 hours 46 minutes) resulted in the opaque suffix:

34PGRBS

The server issuing this suffix was hosted on an computer with IP 150.163.34.243, and accessible from the port 800, leading to the use of the opaque prefix:

8JMKD3MGP8W

In this way, the **identifier for the item** became:

8JMKD3MGP8W/34PGRBS

□

The two types of **identification systems on two levels** are presented below in detail. At first, the **identifier of an item**, displaying the domain name, is called **name of the uniform repository of that item**. In the second type, the **identifier of an item**, built based on the IP, is called **item IBIp**.

## 6 Label construction rules based on domain name

In the **identification system on two levels** presented in this chapter, the **identifier of an item** is also called **name of the uniform repository of that item** because it can be used to define a four directories sequence serving to store in a file system, the **item** being identified, if it is of the type **information item**.

The repositories are called uniform because, through them, any **information item** can be stored on any file system, under a same directory without name conflict when considering other **information items**, thus facilitating the storage of copies in different file systems and still the migration of **information items** between them.

In the **name of the uniform repository of an item**, the prefix and the suffix are separated by `"/"` and each is in turn subdivided into two parts also separated by `"/"`. Thus, the **labels** are constituted of four parts, which can become a sequence of four directories.

As announced, the two parts of the prefix are built from a computer's domain name (hostname) and possibly a port number.

Regarding the suffix, the two parts are constructed from a date and time information expressed in Coordinated Universal Time (UTC).

Thus, the four parts of the **name of the uniform repository of an item** are formed by, in this order:

a) a subdomain name,

b) a domain word[1], and eventually a port number, separated by `"."` or by `"@"`,

c) a year and

d) a month, day, hour, minute, and eventually second[2], separated by `"."`.

These four parts are recognizable in Example 1 of the previous chapter, where the **name of the uniform repository of the item** was:

`sid.inpe.br/mtc-m18@80/2009/02.16.17.46`

[1] A domain name consists of words separated by dots.
[2] or fraction of a second.

To precisely define the syntax of the **name of the uniform repository of an item**, this standard uses a BNF grammar – Backus Normal Form or Backus-Naur Form – (augmented) (CROCKER, 1982; CROCKER; OVERELL, 2008) with the following change: `"|"` is used for alternatives in place of `"/"`.

The syntax of the part relating to the prefix incorporates the rules relative to the formation of "domain name" as defined in Section 3.1 entitled "Name space specifications and terminology" by Mockapetris (1987), and "hostname" as defined in Section 3.2.2 entitled "Server-based Naming Authority" by Berners-Lee et al. (1998).

Table 6.1 contains the rules for the formation of the **name of the uniform repository of an item**.

Table 6.1 - Rule defining the formation of the **name of the uniform repository of an item**

```
 repository  =  prefix "/" suffix
                     ; ex: sid.inpe.br/mtc-m19/2010/08.25.12.38
     prefix  =  subdomain "/" word [("." | "@") port]
                     ; ex: sid.inpe.br/mtc-m19
  subdomain  =  *(word ".") last-word ["."]; ex: dpi.inpe.br
       word  =  ALPHANUM | (ALPHANUM *(ALPHANUM | "-") ALPHANUM); ex: sid
   ALPHANUM  =  ALPHA | DIGIT
      ALPHA  =  LOWALPHA | UPALPHA
   LOWALPHA  =  "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
                "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
                "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
    UPALPHA  =  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
                "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
                "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
      DIGIT  =  "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
                "9"
  last-word  =  ALPHA | (ALPHA *(ALPHANUM | "-") ALPHANUM); ex: br
       port  =  1*DIGIT; ex: 80
     suffix  =  year "/" month "." day "." hour "." minute ["." second]
                     ; ex: 2010/08.25.12.38
       year  =  4*DIGIT; ex: 2010
      month  =  2DIGIT; ex: 08
        dia  =  2DIGIT; ex: 25
       hour  =  2DIGIT; ex: 12
     minute  =  2DIGIT; ex: 38
     second  =  integer ["." fraction]
    integer  =  2DIGIT
   fraction  =  1*DIGIT
```

By adding the `port` to the `word` in the `prefix`, the separator can be symbols `"."` or `"@"`. It is recommended to use just the symbol `"."`. The use of symbol `"@"` was

necessary for the deployments of the **IBI** prior to August 2010. The drawback of this symbol is that it induces certain applications to interpret the **identifier of an item** as an e-mail address.

The rule `subdomain` is denoted `hostname` in Berners-Lee et al. (1998).

The domain name of a computer being insensitive to upper and lower case, this property extends to the **name of the uniform repository of an item**. Thus, `sid.inpe.br/mtc-m18@80/2009/02.16.17.46` and `sid.INPE.br/MTC-m18@80/2009/02.16.17.46` are equivalent. In practice, it is recommended to use only lower case in the prefix generation.

In addition to the syntactic rules of Table 6.1, the **name of the uniform repository of an item** must satisfy the semantic rules defined by Algorithms 1, 2 and 3. In turn, by construction, Algorithm 3 generates a **label** that satisfies the syntactic rules of Table 6.1.

Algorithm 1 is the description of the **label generator** denoted $a$ in Figura 5.1 and used by the **identification system** $f$ for the identification of the **identification subsystems**.

Algorithm 2 is the description of the **label generator** denoted $b$ in Figura 5.1 and used by any **identification subsystem**.

Algorithm 3 is the description of the **label generator** denoted $c$ in Figura 5.1 and used by the **identification system on two levels** $s$.

The **identification system on two levels** works in a distributed manner, a server for each **identification subsystem**. The servers are hosted on computers having domain names, and the access to the servers being done via ports.

Upon receiving a request for identification of an **item** $i$, the server responsible for the **identification subsystem** $g(i)$ executes Algorithms 1, 2 and 3 and returns the **identifier of the item**.

**Algorithm 1** – MountPrefixOfNameOfRepositoryOfAnItem.

| | |
|---|---|
| Input: | `computer` (*string representing the domain name (in lower case) of the computer (possibly virtual) hosting the server responsible for the identification subsystem),*<br>`port` (*decimal integer representing the port number to access the server responsible for the identification subsystem*). |
| Output: | `prefix` (*string*). |
| Auxiliary: | `part` (*integer*),<br>`aux` (*string*),<br>`subdomain` (*string*),<br>`first-word` (*string*),<br>`word-port` (*string*),<br>`c` (*character*). |

```
1.  aux ← computer
2.  part ← 2
3.  While aux ≠ "", do
4.      c ← LeaveQueue(aux)
5.      If c = "." then
6.          parte ← 1
7.      Else
8.          If parte = 1 then
9.              EnterQueue(subdomain, c)
10.         Else
11.             EnterQueue(first-word, c)
12. If port = 80 then
13.     word-port ← first-word
14. Else
15.     word-port ← Concatenate(first-word, ".", port)
16. prefix ← Concatenate(subdomain, "/", word-port)
```

**Algorithm 2** – MountSuffixOfNameOfRepositoryOfAnItem.

| | |
|---|---|
| Input: | `date` (*decimal rational obtained from output of Algorithm 4:* CreateDateToMountSuffixOfIBIOfAnItem), |
| Output: | `suffix` (*string*). |
| Auxiliary: | `year` (*decimal integer*), |
| | `month` (*decimal integer*), |
| | `day` (*decimal integer*), |
| | `hour` (*decimal integer*), |
| | `minute` (*decimal integer*), |
| | `second` (*decimal integer*), |
| | `fraction-of-a-second` (*decimal integer*). |

1. `year` ← Extract(`date`, year)
2. `month` ← Extract(`date`, month)
3. `day` ← Extract(`date`, day)
4. `hour` ← Extract(`date`, hour)
5. `minute` ← Extract(`date`, minute)
6. `second` ← Extract(`date`, second)
7. `fraction-of-a-second` ← Extract(`date`, fraction-of-a-second)
8. `suffix` ← Concatenate(`year`, `"/"`, `month`, `"."`, `day`, `"."`, `hour`, `"."`, `minute`)
9. **If** `fraction-of-a-second` = `"0"` **then**
10. ⎸ **If** `second` ≠ `"00"` **then**
11. ⎸ ⎸ `suffix` ← Concatenate(`suffix`, `"."`, `second`)
12. **Else** `suffix` ← Concatenate(`suffix`, `"."`, `second`, `"."`, `fraction-of-a-second`)

**Algorithm 3** – MountNameOfRepositoryOfAnItem.

INPUT: `prefix` (*string obtained from output of Algorithm 1:* MountPrefixOfNameOfRepositoryOfAnItem),
`suffix` (*string obtained from output of Algorithm 2:* MountSuffixOfNameOfRepositoryOfAnItem).
OUTPUT: `repositório` (*string*).

*1.* `repositório` ← Concatenate(`prefix`, `"/"`, `suffix`)

The algorithms presented above use the following routines.

Routine Concatenate concatenate the strings informed in its arguments.

Routine EnterQueue add on the right side of the string informed in the first argument, one more character, the one informed in the second argument.

Routine LeaveQueue takes the first character of the string informed its argument, and returns this character.

Routine Extract returns, in the format compatible with the rules of Table 6.1, the decimal number of the unit informed in the second argument, when the date in second or fraction of a second, informed in the first argument, is converted to Coordinated Universal Time (UTC).

The time standard UTC is used in order to allow the continued operation of **identification system on two levels** even in case of entering/leaving Daylight saving time or exchanging of computers at different longitudes, and running a same **identification subsystem**.

Algorithm 1 separates the first word of the domain name [3] of the computer (possibly virtual) from the subdomain without this word, thus dividing the prefix into two parts, the first part containing the subdomain, and the second, the first word.

According to Algorithm 1, when the port number is 80, it is omitted in the second part of the prefix. As port 80 is the port usually used by HTTP servers running the CGI scripts that implement the algorithms 1, 2 and 3, the prefix of the **names of the uniform repository of an item** is, in this way, usually shorter.

[3] A domain name consists of words separated by dots.

Algorithm 2 generates a **label** (suffix) based on the value of the `date` provided by the **temporal distributor** (see output `date-for-suffix` of Algorithm 4).

In case a faster response from the **identification subsystem** is needed, a smaller granularity $r$ can be chosen. However, the possibility to meet identification requests by using a large number of **identificaton subsystems** (remembering that the prefix granularity is extremely thin) is another solution to mitigate the problem of a high frequency of requests.

Algorithm 3 concatenates the prefix and suffix and interposed between them the symbol `"/"`. The presence, in this position, of a symbol that does not belong to the alphabets used in generating the prefix and suffix, makes the concatenation reversible because, with its presence, it is possible to recognize unambiguously, the suffix and prefix after the concatenation.

For the correct operation, the inputs of Algorithm 1 should be: the domain name (hostname) of the computer or virtual computer (virtual host) that hosts the server responsible for **identification subsystem**, and the port which gives access to that server.

The pair formed by the domain name (in lower case) of the computer and the access port for the **identification subsystem** $g(i)$ is interpreted in Figure 5.1 as the output, in $C^2$, of the so-called **installer** $e$, receiving as input, in $S$, the **identification subsystem** $g(i)$.

The computer's domain name (hostname) can be obtained, for example, through the `nslookup` command.

As regards Algorithm 2, the input should be the date $t(i)$, interpreted in Figura 5.1 as the output, in $T_r$, of a **temporal distributor**, receiving as input, in $g^*(g(i))$, the **item** $i$ itself.

For any $i \in I$, the role of the **temporal distributor** used by the **identification subsystem** $g(i)$ is to distribute within the grid $T_r$, with granularity of $r$ seconds, the identification requests related to the **items** of the set $g^*(g(i))$.

The **temporal distributor** is described in detail below, where $I'$ represents the domain $g^*(g(i))$ of a given **identification subsystem** $g(i)$.

Let $I'$ be a set of **items**.

Let $\mathbb{Q}^+$ be the set of positive rational numbers.

Let $\mathbb{R}^+$ be the set of positive real numbers.

Let $G \triangleq \{60, 1, 1/10, 1/100, ...\}$ be the subset of rational numbers, defining the possible temporal granularities: minute, seconds and fractions of a second.

Let $t_i \in \mathbb{R}^+$ be the date, expressed in fraction of a second[4], of the request of the identification of the **item** $i \in I'$ (it is assumed that $i \mapsto t_i$ is injective).

Let $[i] \in \{1, 2, ..., |I'|\}$ be the value indicating that the **item** $i \in I'$ was the $[i]^{\text{th}}$ **item** to request an identification, i.e., $[i]$ is given by:

$$[i] \triangleq \sum_{j \in I'} \begin{cases} 1 & \text{if} \quad t_j \leq t_i, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for any } i \in I'.$$

Let $]k[ \in I'$ be the value indicating that the $k^{\text{th}}$ **item** to request an identification is the **item** $]k[$, i.e., $]k[$ is given by:

$$]k[ \triangleq i \;\Leftrightarrow\; k = [i], \qquad \text{for any } k \in \{1, 2, ..., |I'|\} \text{ and } i \in I'.$$

Let $t^{(r)} \in \mathbb{Q}^+$ be the date $t$ rounded in $r$ seconds, i.e, $t^{(r)}$ is given by:

$$t^{(r)} \triangleq r\text{int}(t/r), \qquad \text{for any } t \in \mathbb{R}^+ \text{ and } r \in G.$$

Let $T_r = \{t^{(r)} : t \in \mathbb{R}^+\}$ be the set of dates rounded in $r$ seconds, being $r \in G$.

Let $t'_i \in T_r$, with $r \in G$ and $i \in I'$, the date, rounded in $r$ seconds, given by:

$$t'_i \triangleq \begin{cases} {t_i}^{(r)} & \text{if } [i] = 1, \\ \max(t(][i] - 1[) + r, {t_i}^{(r)}) & \text{if } [i] = 2, ..., |I'|. \end{cases}$$

In the above expression, $t(][i] - 1[)$ is the date provided by the **temporal distributor** referring to the **item** $][i] - 1[$ immediately preceding the **item** $i$, considering the identification request dates. The date $t'_i$ will be considered the date on which the **temporal distributor** provides its answer $t(i)$ used in the generation of the **label** used to identify the **item** $i$. In other words, $t'_i$ will be considered the date of generation of the **label** of the **item** $i$, while $t(i)$ is the date used by the **label generator** $b$ to mount the suffix of the identifier of $i$.

[4]More precisely in Unix or POSIX time.

Let $t'(i) \in {T_r}^2$, with $r \in G$ and $i \in I'$, be the pair of dates, rounded in $r$ seconds, given by:

$$t'(i) \triangleq \begin{cases} (t'_i, t'_i - r) & \text{if } [i] = 1, \\ (t'_i, t(]{[i] - 1}[)) & \text{if } [i] = 2, ..., |I'|. \end{cases}$$

The mapping $i \mapsto t'(i)$ defines a function $t'$ from $I'$ to ${T_r}^2$.

Let $r_{(t,s)} \in G$, with $t > s$, be the greater granularity $r$ such that the date $t$ rounded in $r$ seconds, is greater than the date $s$, i.e., $r_{(t,s)}$ is given by:

$$r_{(t,s)} \triangleq \max(\{r \in G : t^{(r)} > s\}), \qquad \text{for any } t \text{ and } s \in \mathbb{R}^+ \text{ such that } t > s.$$

Note that $r_{(t,t-r)} = r$ for any $t \in \mathbb{R}^+$ and $r \in G$.

Let $t(i) \in T_{r_{t(i)}} \subset T_r$, with $r \in G$, be the date $t'_i$ rounded in $r_{t'(i)}$ seconds:

$$t(i) \triangleq {t'_i}^{(r_{t'(i)})}, \qquad \text{for any } i \in I'.$$

Note that, for any $r \in G$, $t(]1[) = {t'_{]1[}}^{(r)}$, and that $r_{t(i)} \leq r$, $T_{r_{t(i)}} \subset T_r$ and $t(]{[i] - 1}[) < t(i) \leq t'_i$, for any $i \in I'$ tal que $[i] > 1$.

A **temporal distributor**, with granularity $r \in G$, is the function $t : I' \to T_r$ such that $i \mapsto t(i)$.

Table 6.2 contains the data of a working example of a **temporal distributor** with granularity of one second ($r = 1$).

Table 6.2 - Exemplification of running a **temporal distributor** with granularity of one second

| $i$ | $t_i$ (seconds) | $[i]$ | $t'(i)$ | $r_{t'(i)}$ | $t(i)$ | suffix |
|---|---|---|---|---|---|---|
| d | 1287587646,394023 | 1 | (1287587646, 1287587645) | 1 | 1287587646 | 2010/10.20.15.14.06 |
| b | 1287588012,2930 | 2 | (1287588012, 1287587646) | 60 | 1287588000 | 2010/10.20.15.20 |
| a | 1287588115,186234 | 3 | (1287588115, 1287588000) | 60 | 1287588060 | 2010/10.20.15.21 |
| c | 1287588115,3462 | 4 | (1287588115, 1287588060) | 1 | 1287588115 | 2010/10.20.15.21.55 |
| g | 1287588115,99623 | 5 | (1287588116, 1287588115) | 1 | 1287588116 | 2010/10.20.15.21.56 |
| f | 1287588116,72 | 6 | (1287588117, 1287588116) | 1 | 1287588117 | 2010/10.20.15.21.57 |
| e | 1287588539,788342 | 7 | (1287588539, 1287588117) | 60 | 1287588480 | 2010/10.20.15.28 |

In Table 6.2, the lines were sorted by increasing dates of identification request (column $t_i$). The values of $[i]$ indicate the order of requests. It is observed, for exam-

ple, that the **item** `g`, despite having requested his identification on a date prior to 1287588116 seconds, received as date $t(i)$, to be used in generating the **label**, the value 1287588116 seconds. This occurred because the **items** `a`, `c` e `g` did, the three, the identification request within one second period. Consequently, the **item** `g` will have to wait for the **items** `a` e `c` first receiving their identification in order to receive its. This "delay" is spreading to the next **item** `f` who made his identification request just the next second (1287588116) to that of **item** `g` (1287588115).

In practice, the **temporal distributor** may be implemented using, for example, the concepts of waiting room and timer, in order to associate to each **item** $i$ a date of identification request $t_i$ (date of the beginnning of the service after waiting in the waiting room) and to turn it the date $t(i)$ within a temporal grid of granularity $r$, for generating the **label**.

In this case, the **items** to be identified by a given **identification subsystem** enters a waiting room. When the server responsible for **identification subsystem** is ready to identify a new **item**, one of the **items** in the waiting room is drawn. The moment of the draw becomes the date of the drawn item identification request.

Algorithm 4 shows how to create, in practice, the date $t(i)$ used, by the **label generator** $b$, to mount the suffix of the identifier of an **item** $i \in I'$.

To create the date $t(i)$, this algorithm has to be executed at the time $t_i$ (date of the request identification), and receive this date $t_i$ as input.

In Algorithm 4 the input, denoted `current-date`, is the date $t_i$, and the output, denoted `date-for-suffix`, is the date $t(i)$.

**Algorithm 4** – CreateDateToMountSuffixOfIBIOfAnItem.

| | |
|---|---|
| Input: | `current-date` (*decimal rational representing the request date of an identification, i.e., the date of running the algorithm itself*). |
| Output: | `date-for-suffix` (*decimal rational*). |
| Global: | `last-date-for-suffix` (*decimal rational representing the date created for the suffix on the previous run the algorithm itself*), `r` (*decimal rational representing the temporal granularity*). |
| Auxiliary: | `rounded-date` (*decimal rational*), `creation-date` (*decimal rational*), `delay` (*decimal rational*), `s` (*decimal rational*), `t` (*decimal rational*), `short-date` (*decimal rational*). |

```
1.  rounded-date ← r * Int(current-date/r)
2.  If last-date-for-suffix doesn't exist then
3.      last-date-for-suffix ← rounded-date − r
4.  last-date-for-suffix ← r * Int(last-date-for-suffix/r)
5.  creation-date ← Max(last-date-for-suffix + r, rounded-date)
6.  delay ← creation-date − current-date
7.  If delay > 0 then Wait(delay)
8.  short-date ← creation-date
9.  s ← r
10. While last-date-for-suffix < short-date do
11.     s ← 10 * s
12.     If s = 10 then s ← 60
13.     t ← short-date
14.     If s > 60 then Break
15.     short-date ← s * Int(creation-date/s)
16. date-for-suffix ← t
17. last-date-for-suffix ← t
```

The global variable `last-date-for-suffix` has the role of a memory that retains the last value of `date-for-suffix` to be reused in creating the next suffix. If the variable `last-date-for-suffix` does not exist (for example, on the first identification request), its value is chosen equal to the current date rounded in $r$ seconds (`rounded-date`) , minus the current granularity $r$ (Lines 2 and 3).

The global variable `r` defines the temporal granularity to be used for the suffix

creation. Its value may be 60 (for the granularity of one minute), 1 (for second) or 0.1 (for tenths of seconds) 0.01 (for hundredth of seconds), 0.001 (for thousandth of seconds), ...

The value of `r` can be changed externally to the algorithm to meet the new conditions of use of the **identification subsystem**.

On Line 1, the `current-date` is rounded in $r$ seconds through the routine INT which returns the integer part of the rational value of its argument.

Line 4 is to reformat the value of the variable `last-data-for-suffix` where there has been a change in the amount of $r$ granularity between two successive suffix creations.

In Line 5, the `creation-date` is calculated by the routine MAX which returns the greater its two arguments. This auxiliary variable is the date $t'_i$. For that date to be the date of the suffix creation, a delay is introduced in Line 7, through the routine WAIT waiting for the time specified in the variable `delay`, every time that the calculated `creation-date` is greater than the `current-date`.

Lines 8 to 16 are used to compute the output `date-for-suffix` that may be a lesser date than the `creation-date` if a shorter version of that date is compatible with `last-data-for-suffix`.

## 7 Label construction rules based on IP

In the **identification system on two levels** presented in this chapter, the **identifier of an item** is opaque, built based on IP, and called **IBIp of that item**.

In the **IBIp of an item**, the prefix and suffix are also separated by `"/"`.

As announced, the prefix is built based on the information of a computer IP and possibly a port number. Regarding the suffix, it is constructed from a date and time information as in the previous chapter. The opacity is obtained by encoding this information.

The concatenation of the prefix and suffix are recognizable in Example 2 of Chapter 5 where the **IBIp of the item** was:

`8JMKD3MGP8W/34PGRBS`

Table 7.1 contains the rules for the formation of the **IBIp of an item**.

Table 7.1 - Rules defining the formation of the **IBIp of an item**

```
    IBIp  =  word "/" word; ex: 8JMKD3MGP7W/385N5PE
    word  =  1*ALFANUM; ex: 385N5PE
ALPHANUM  =  ALPHA | DIGIT
   ALPHA  =  LOWALPHA | UPALPHA
LOWALPHA  =  "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
             "j" | "k" | "l" | "m" | "n" | "p" | "q" | "r" |
             "s" | "t" | "u" | "w" | "x"
 UPALPHA  =  "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
             "J" | "K" | "L" | "M" | "N" | "P" | "Q" | "R" |
             "S" | "T" | "U" | "W" | "X"
   DIGIT  =  "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

From Table 7.1, it is observed that the characters `"0"`, `"O"`, `"1"`, `"I"`, `"V"`, `"Y"` e `"Z"` were excluded. The first five to remove any doubt in the reading of these characters when one makes use of certain fonts. The last two were reserved if it is necessary to define in the future a case sensitive identification system.

As with the **name of the uniform repository of an item** of the previous chapter, the **IBIp of an item** is case insensitive. So `8JMKD3MGP8W/34PGRBS` and `8jmkd3mgp8w/34pgrbs` are equivalent. In practice, it is recommended to use only uppper case characters for the **label** generation.

In addition to the syntactic rules of Table 7.1, an **IBIp of an item** must satisfy the semantic rules defined by the algorithms 5, 6 and 7. In turn, by construction, Algorithm 7 generates a **label** that satisfies the syntax rules of Table 7.1.

Algorithm 5 is the description of the **label generator** denoted $a$ in Figure 5.1 and used by the **identification system** $f$ for the identification of the **identification subsystems**.

Algorithm 6 is the description of the **label generator** denoted $b$ in Figure 5.1 and used by any **identification subsystems**.

Algorithm 7 is the description of the **label generator** denoted $c$ in Figure 5.1 and used by the **system for IBI generation** $s$.

**Algorithm 5** – MountPrefixOfIBIpOfAnItem.

| | |
|---|---|
| Input: | `IP` (*string representing the IP of the computer hosting the server responsible for the identification subsystem*), |
| | `port` (*decimal integer representing the port number to access the server responsible for the identification subsystem*). |
| Output: | `prefix` (*string*). |
| Global: | `decim-to-IBIp` (*conversion Table 7.6 from decimal to IBIp*), |
| | `decim-to-IPv4` (*conversion Table 7.4 from decimal to IPv4*), |
| | `decim-to-IPv6` (*conversion Table 7.5 from decimal to IPv6*). |
| Auxiliary: | `decim-coded-IP` (*decimal integer*), |
| | `coded-IP` (*string*), |
| | `coded-port` (*string*). |

*1.* **If** `port` $= 800$ **then**
*2.* `coded-port` $\leftarrow$ `""`
*3.* **Else**
*4.* `coded-port` $\leftarrow$ ConvertFromDecimal(`port`, `decim-to-IBIp`)
*5.* **If** `"."` $\in$ `IP` **then**
*6.* `decim-coded-IP` $\leftarrow$ ConvertToDecimal(`IP`, `decim-to-IPv4`)
*7.* `coded-IP` $\leftarrow$ ConverterDeDecimal(`decim-coded-IP`, `decim-to-IBIp`)
*8.* `prefix` $\leftarrow$ Concatenate(`coded-IP`, `"W"`, `coded-port`)
*9.* **Else**
*10.* `decim-coded-IP` $\leftarrow$ ConvertToDecimal(`IP`, `decim-to-IPv6`)
*11.* `coded-IP` $\leftarrow$ ConvertFromDecimal(`decim-coded-IP`, `decim-to-IBIp`)
*12.* `prefix` $\leftarrow$ Concatenate(`coded-IP`, `"X"`, `coded-port`)

**Algorithm 6** – MountSuffixOfIBIpOfAnItem.

| | |
|---|---|
| Input: | `date` (*decimal rational obtained from output of Algorithm 4:* CreateDateToMountSuffixOfIBIOfAnItem), |
| Output: | `suffix` (*string*). |
| Global: | `decim-to-IBIp` (*conversion Table 7.6 form decimal to IBIp*). |
| Auxiliary: | `integer-part` (*decimal integer*), |
| | `second` (*decimal integer*), |
| | `fraction` (*decimal integer*), |
| | `coded-fraction` (*string*). |

1. `integer-part` ← Int(`date`)
2. `second` ← `integer-part` − 807235200
3. `fraction` ← `date` − `integer-part`
4. `suffix` ← ConvertFromDecimal(`second`, `decim-to-IBIp`)
5. **If** `fraction` ≠ 0 **then**
6. `coded-fraction` ← ConvertFromDecimal(`fraction`, `decim-to-IBIp`)
7. `suffix` ← Concatenate(`suffix`, `"W"`, `coded-fraction`)

**Algorithm 7** – MountIBIpOfAnItem.

| | |
|---|---|
| Input: | `prefix` (*string obtained from output of Algorithm 5:* MountPrefixOfIBIpOfAnItem), |
| | `suffix` (*string obtained from output of Algorithm 6:* MountSuffixOfIBIpOfAnItem). |
| Output: | `IBIp` (*IBIp of an item*). |

1. `IBIp` ← Concatenate(`prefix`, `"/"`, `suffix`)

The algorithms presented above use the following routines.

Routine ConvertToDecimal converts, from a specific numbering system to the decimal numbering system, the string, informed in the first argument, to a decimal integer, according to the table reported in the second argument. This conversion uses the inverse table of the specified table.

Some examples of use of routine ConvertToDecimal are shown in Table 7.2.

Routine ConvertFromDecimal converts, from the decimal number system to another number system, the decimal integer, informed in the first argument, to a

Table 7.2 - Examples of conversion using Routine ConvertToDecimal

| IP | table | output |
|---|---|---|
| 150.163.2.174 | `decim-to-IPv4` | 4588904456580 |
| 2001:252:0:1::2008:6 | `decim-to-IPv6` | 478239719325051908572237 |

string, according to the table reported in the second argument.

Some examples of use of routine ConvertFromDecimal are shown in Table 7.3.

Table 7.3 - Examples of conversion using Routine ConvertFromDecimal

| decimal | table | output |
|---|---|---|
| 1 | `decim-to-IBIp` | 3 |
| 19050 | `decim-to-IBIp` | U5H |
| 480992662 | `decim-to-IBIp` | 38G3TS3 |
| 4588904456580 | `decim-to-IBIp` | J8LNKAN8P |
| 478239719325051908572237 | `decim-to-IBIp` | 7URMDHLL9SSN2D89M |

The input of Algorithm 5 is the same as that of Algorithm 1.

In Lines 1 to 4 of Algorithm 5, the port number is tested to see if it is equal to 800, in this case there is no conversion of this number and an empty string is returned, otherwise the port number is encoded using Table 7.6.

Unlike the previous chapter, the port number 80 is not considered because more than one virtual host may exist using the same port 80. In tests of this standard for an identification system, port numbers like 800 and 802 were used for accessing the scripts that implement the **label generators** used by the **identification systems on two levels** hosted on different virtual hosts, but within the same real computer, so associated with same IP. For the case where there is only one virtual host, the port number 800 was adopted. In this way, each time the port number 800 is used, the prefix of the **IBIp of an item** becomes shorter.

In Line 5 of Algorithm 5 the type of IP is tested. If the IP is of type IPv4, Line 8, the character `"W"` is used in the concatenation to separate the encoded IP from the encoded port number. If the IP is of type IPv6, on Line 12 the character `"X"` is used for this purpose.

Because the characters `"W"` and `"X"` are not part of the graphemes of Table 7.6, it is possible, if necessary, to decode the prefix of the **IBIp of an item**.

To encode an IP, it is considered that its value represents a number within a specific numbering system. The encoding consist then to convert the representation in the original system to a representation in another numbering system, denoted here IBIp. As there are two types of IP: IPv4 and IPv6, it is considered two original numbering systems, denoted respectively IPv4 and IPv6.

The conversion of an IP of one of those two numbering systems to the IBIp system is made in Algorithm 5, using their representation in the decimal system.

Thus, to convert a representation in the IPv4 (respectively IPv6) system to its representation in the IBIp system, first the conversion of the representation in the IPv4 (respectively IPv6) system to its representation in the decimal system based on the inverse table of Table 7.4 (respectively, 7.5) is made, and then the conversion of its representation in the decimal system to its representation in the IBIp system based on Table 7.6.

The input of Algorithm 6 is the same as that of Algorithm 2.

In Line 2, Algorithm 6 calculates the difference in seconds between the variable `integer-part` and the constant: 807235200. This constant is the number of seconds (in Unix time) corresponding to the date 19950801T000000Z (date in ISO 8601 format) of the beginning of the month during which the first generation of a **label** was made following this standard. This operating mode, allow to generate shorter **labels**, since obtained as the result of the conversion of smaller numbers.

In Line 7, the character `"W"` is used to separate the coded integer part from the fractional coded part. Because the character `"W"` is not part of the graphemes of Table 7.6, it is possible, if necessary, to decode the prefix of the **IBIp of an item**.

Table 7.4 - Conversion table from decimal to IPv4

| decimal | IPv4 |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | . |

Table 7.5 - Conversion table from decimal to IPv6

| decimal | IPv6 |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | a |
| 11 | b |
| 12 | c |
| 13 | d |
| 14 | e |
| 15 | f |
| 16 | : |

Table 7.6 - Conversion table from decimal to IBIp

| decimal | IBIp |
|---|---|
| 0 | 2 |
| 1 | 3 |
| 2 | 4 |
| 3 | 5 |
| 4 | 6 |
| 5 | 7 |
| 6 | 8 |
| 7 | 9 |
| 8 | A |
| 9 | B |
| 10 | C |
| 11 | D |
| 12 | E |
| 13 | F |
| 14 | G |
| 15 | H |
| 16 | J |
| 17 | K |
| 18 | L |
| 19 | M |
| 20 | N |
| 21 | P |
| 22 | Q |
| 23 | R |
| 24 | S |
| 25 | T |
| 26 | U |

## REFERENCES


BERNERS-LEE, T.; FIELDING, R.; IRVINE, U. C.; MASINTER, L. **Uniform Resource Identifiers (URI): Generic syntax**. Washington DC: The Internet Engineering Task Force (IETF), Aug. 1998. 40 p. RFC 2396. Disponível em: <http://tools.ietf.org/html/rfc2396>. Acesso em: 19 ago. 2010. 14, 15

CROCKER, D. H. **Standard for the format of ARPA Internet messages**. Washington DC: The Internet Engineering Task Force (IETF), Aug. 1982. 47 p. RFC 822. Disponível em: <http://tools.ietf.org/html/rfc822>. Acesso em: 19 ago. 2010. 14

CROCKER, D. H.; OVERELL, P. **Augmented BNF for Syntax Specifications: ABNF**. Washington DC: The Internet Engineering Task Force (IETF), Jan. 2008. 16 p. RFC 5234. Disponível em: <http://tools.ietf.org/html/rfc5234>. Acesso em: 19 ago. 2010. 14

MOCKAPETRIS, P. **Domain names - concepts and facilities**. Washington DC: The Internet Engineering Task Force (IETF), Nov. 1987. 55 p. RFC 1034. Disponível em: <http://tools.ietf.org/html/rfc1034>. Acesso em: 19 ago. 2010. 14

## APPENDIX A - FIBER DEFINITION

Let $X$ and $Y$ two non-empty sets, and $f$ a mapping from $X$ to $Y$.

The *image of a subset $A$ of $X$ through $f$*, is the subset of $Y$ denoted by $f(A)$ and given by:

$$f(A) \triangleq \{f(x) : x \in A\}, \qquad \text{for any } A \subset X.$$

The *inverse image of a subset $B$ of $Y$ through $f$*, is the subset of $X$ denoted by $f^{-1}(B)$ and given by:

$$f^{-1}(B) \triangleq \{x \in X : f(x) \in B\}, \qquad \text{for any } B \subset Y.$$

The *fiber of $y$ through $f$* is the subset of $X$ denoted by $f^*(y)$ and given by:

$$f^*(y) \triangleq f^{-1}(\{y\}), \qquad \text{for any } y \in f(X).$$