IBPCA: IBP with MD5

Rebecca Collins
rcollins@cs.utk.edu

Abstract

This document presents a description of the five client applications for the Content Addressable IBP (IBPCA). The description includes the data structures and calls used by the C implementation of IBPCA. Knowledge of IBP is assumed.

Introduction

IBPCA uses MD5 hashes to reference data stored in the depots. This is done by incorporating the MD5 hash of stored data in the read capability for a storage area. When new data is appended to a storage area, the updated storage area requires a new read capability. IBPCA operates a level above IBP.

Contents

1. DATA STRUCTURES

1.1 IBP_depot, IBP_timer, and IBP_attributes

IBP_depot, IBP_timer, and IBP_attributes are all defined in IBP_ClientLib.h, a library from the IBP. Descriptions of IBP_depot, IBP_timer, and IBP_attributes can be found in section 1 of IBP v1.1.1 API.

1.2 ulong_t

ulong_t is an unsigned long integer. It is also defined in IBP_ClientLib.h

1.3 IBP_cap

IBP_cap is a char * . It is also defined in IBP_ClientLib.h. IBP_cap's are supposed to have a special format. With IBP, they have the format:

```
ibp://hostname:port/key/WRMKey/WRM
```

There are read, write, and management capabilities with IBP, but with IBPCA there are only read and write capabilities. IBPCA capabilities have this format:

```
Read capabilities:
ibpca://hostname:port/R/MD5-checksum
Write capabities:
ibpca://hostname:port/W/random_string
```

where

- **hostname** and **port** are the same as in the IBP capability
- **MD5-checksum** is the MD5 hash of the data in the storage depot that the read capability references
- **random_string** is a randomly generated string of length 32

1.4 ibp_probe_info

ibp_probe_info is used in **IBP_CA_manage**.

Variable	Type

size	int
maxSize	int
exists	int
attrib	struct ibp_attributes

When **IBP_CA_manage** is being used with the IBP_CA_MANAGE_PROBE command, the values in the **ibp_probe_info** struct will be filled in with the correct values upon return. Their initial values do not matter.

When **IBP_CA_manage** is being used with the IBP_CA_MANAGE_TIME command, the the duration of the storage will be extended to **attrib.duration** if it is shorter than **attrib.duration**. Nothing happens otherwise. The values of **maxSize**, **exists**, and **attrib** will be updated upon return.

When **IBP_CA_manage** is being used with the IBP_CA_MANAGE_SIZE command, the size of the storage will be increased to **maxSize** if the current size is smaller than **maxSize**. Nothing happens otherwise. The values of **maxSize**, **exists**, and **attrib** will be updated upon return.

When **IBP_CA_manage** is being used with the IBP_CA_MANAGE_DEL command, the values of the **ibp_probe_info** struct are ignored.

2. CLIENT APPLICATIONS

Five client applications are implemented for the IBPCA:

int **IBP_CA_allocate**(IBP_depot depot, IBP_timer timeout, ulong_t maxsize, IBP_attributes attributes, IBP_cap writecap);

int **IBP_CA_load**(IBP_cap ca_readcap, IBP_timer timeout, char *buf, ulong_t size, ulong_t offset);

int **IBP_CA_manage**(IBP_cap man_cap, IBP_timer timeout, int cmd, ibp_probe_info *status);

int **IBP_CA_store**(IBP_cap ca_writecap, IBP_timer timeout, char *data, ulong_t size, IBP_cap readcap);

int **IBP_CA_store_block**(IBP_depot depot, IBP_timer timeout, ulong_t size, char *data, IBP_attributes attributes,

```
                        IBP_cap ca_readcap);
```

2. 1 IBP_CA_allocate

	variable name	**variable type**
parameter	depot	IBP_depot
	timeout	IBP_timer
	maxsize	ulong_t
	attributes	IBP_atrributes
(output)	writecap	IBP_cap
Return value		void *

IBP_CA_allocate allocates **maxsize** bytes of storage into the **depot**, with attributes **attributes**. The duration must be finite, and the data type must be byte-array.

Return Values
On successful completion, 0 is returned, otherwise –1 is returned and an error message is sent to stderr. The following conditions will cause IBP_CA_allocate to fail:

- Invalid attributes – For example, infinite duration
- IBP_allocate error – Described in section 3 of IBP v1.1.1 API.

2.2 IBP_CA_load

	variable name	**variable type**
parameter	readcap	IBP_cap
	timeout	IBP_timer
	buf	char *
	size	ulong_t
Return value		ulong_t

IBP_CA_load loads **size** bytes, starting at the **offset** position, from the byte-array accessed through **readcap** and stores the bytes into **buf**.

Return Values
On successful completion, the number of bytes read is returned, otherwise –1 is returned and an error message is sent to stderr. The

following conditions will cause IBP_CA_load to fail:

- Invalid size/offset
- Readcap has an expired duration
- IBPCA internal error
- IBP_load error – Described in section 4 of IBP v1.1.1 API.

2.3 IBP_CA_manage

	variable name	variable type
parameter	man_cap	IBP_cap (a readcap or a writecap)
	timeout	IBP_timer
	cmd	int
	status	ibp_probe_info *
Return value		void

IBP_CA_manage lets the user perform the following operations on a storage area (**IBP_MANAGE_SIZE** and **IBP_CA_MANAGE_DEL** require a writecap):

- **IBP_CA_MANAGE_PROBE** updates **status** with information about the storage area: whether it exists (if the capability is valid), its attributes, size, and maximum capacity.
- **IBP_CA_MANAGE_TIME** extends the duration of a storage area unless the new duration is less than the one that already exists.
- **IBP_CA_MANAGE_SIZE** increases the maximum size of a storage area unless the new size is smaller than the one that already exists.
- **IBP_CA_MANAGE_DEL** deletes a write capability

Return Values
On successful completion, 0 is returned, otherwise –1 is returned and an error message is sent to stderr. The following conditions will cause IBP_CA_manage to fail:

- IBPCA internal error
- Storage area does not exist. That is, writecap/readcap is invalid or expired
- A readcap was sent when a writecap was required
- IBP_manage error – Described in Section 7 of IBP v1.1.1 API.

2.4 IBP_CA_store

	variable name	**variable type**
parameter	writecap	IBP_cap
	timeout	IBP_timer
	data	char *
	size	ulong_t
(output)	readcap	IBP_cap
Return value		ulong_t

IBP_CA_store appends to a **writecap** previously obtained from **IBP_CA_allocate**. The first **size** bytes of **data** are appended to the byte-array referenced by **writecap**. **Readcap** is then constructed from the entire byte-array's MD5-checksum.

Return values
On successful completion, 0 is returned, otherwise –1 is returned and an error message is sent to stderr. The following conditions will cause IBP_CA_store to fail:

- Expired or Invalid writecap
- IBPCA internal error
- IBP_store error –Described in section 3 of IBP v1.1.1 API.

2.5 IBP_CA_store_block

	variable name	**variable type**
parameter	depot	IBP_depot
	timeout	IBP_timer
	size	ulong_t
	data	char *
	attributes	IBP_attributes
(output)	readcap	IBP_cap
Return value		int

IBP_CA_store_block allocates and stores **data** into the **depot**. The duration of the storage must be finite, and the data type must be byte-array. The allocated storage space will be read only. If the data to be stored is identical to one already stored, it will not be sent over the network a second time. Instead, the readcap for the original storage will be returned and the duration of the storage will be updated.

Return values
On successful completion, 0 is returned, otherwise –1 is returned and an error message is sent to stderr. The following conditions will cause IBP_CA_store_block to fail:

- Invalid attributes – For example, infinite duration
- IBP_store error – Described in section 3of IBP v1.1.1 API.
- IBP_allocate error – Described in section 2 of IBP v1.1.1 API.

3. INPUT/OUTPUT SUMMARY

IBPCA calls IBP commands to access the IBP depots. The capabilities are treated a bit differently in the IBPCA since there are no manage capabilities and the read capabilities are updated after every store. The following table summarizes the IBP commands called by IBPCA client applications and the role of capabilities in the procedures.

IBPCA **command**	IBP **command**	IBP **Requires**	IBPCA **Requires**	IBP **Returns**	IBPCA **Returns**
IBP_CA_allocate	IBP_allocate			IBP readcap writecap managecap	IBP_CA writecap
IBP_CA_store	IBP_store	IBP writecap	IBP_CA writecap		IBP_CA readcap
IBP_CA_manage	IBP_manage	IBP managecap	IBP_CA writecap **or** readcap		
IBP_CA_store_block	IBP_allocate IBP_store				IBP_CA readcap

4. ADVANTAGES OF USING IBPCA

MD5 is a message-digest algorithm that generates a unique 16 byte string for a data block. This string is used as a checksum in the IBPCA. Changing one bit of the data will change the MD5 checksum that is computed from the data. It is possible to have two files with the same checksum, but it is highly unlikely to happen at random. It is currently computationally infeasible to deliberately create a file with a given checksum or two files with the same checksum [1]. Since there is reasonable assurance that two files will have different checksums, the checksums can be used to distinguish files. The properties of the MD5 hashes give the IBPCA several advantages over the original IBP.

4.1 Checking Integrity of Stored Data

The client can check the integrity of stored data since the MD5 checksum of a file should be the same before and after the file is stored in the IBP depots. The readcap has the original checksum in it, and the checksum of the downloaded data is easily computed.

4.2 Preventing Redundant Network Traffic

When a client wants to store a file, the checksum of the file can be compared to the checksums of files that are already stored. If the file is already stored, then the client just gets access to the existing storage and the data is not transferred a second time. This saves bandwidth and disk space on the depot. IBP_CA_store_block is the only client application that currently implements this feature.

References

[1] Rivest, R. “The MD5 Message-Digest Algorithm”, RFC 1321, MIT and RSA Data Security, Inc., April 1992.