



## **icare\_f77\_hdf\_tools**

**Outils Fortran 77 pour la manipulation des fichiers HDF  
et de leurs Scientific Datasets**

**Manuel d'utilisation**

**Version 1.0**



**Centre de Gestion et de  
Traitement de Données**

Réf. : 0803001-NT-UDEV-V01-R00

Rév. : 1.0      Date : 06/03/2008

Page : 2 / 18

	Nom et Fonction	Date	Signature
Préparé par	<b>SIX Bruno CGTD ICARE</b>	<b>06/03/2008</b>	

**DIFFUSION**

**DIFFUSION INTERNE**

CGTD

**DIFFUSION EXTERNE**

Libre



*Historique et révisions*

06/03/2008	v1.0.0 : Création
------------	-------------------



## **Table des matières**

<b>Introduction .....</b>	<b>5</b>
<b>Compilation.....</b>	<b>5</b>
<b>Utilisation .....</b>	<b>5</b>
<b>Note sur la calibration.....</b>	<b>6</b>
<b>Fonction HDFSINFO .....</b>	<b>7</b>
<b>Fonction HSDSINFO.....</b>	<b>8</b>
<b>Fonction HGSDSRC .....</b>	<b>9</b>
<b>Fonctions HGSDSRU et HGSDSRUS.....</b>	<b>10</b>
<b>Fonction HGSDSUC .....</b>	<b>11</b>
<b>Fonctions HGSDSUU et HGSDSUUS.....</b>	<b>13</b>
<b>Fonction HGSDSDC .....</b>	<b>15</b>
<b>Fonctions HGSDSDU et HGSDSDUS.....</b>	<b>17</b>



## Introduction

La bibliothèque ICAF77HDF contient un certain nombre de fonctions permettant d'obtenir des informations sur un fichier HDF ou sur les SDS (Scientific DataSets) de ce fichier, ainsi que d'extraire de manière plus ou moins générique le contenu des SDS de ce fichier.

Le langage Fortran 77 ne permettant dynamiquement ni allocation, ni typage, ni dimensionnement, plusieurs fonctions sont fournies pour l'extraction, dans un ordre croissant de fonctionnalités mais décroissant d'efficacité.

Le choix de la fonction à utiliser dépend essentiellement de connaissance que l'on a du SDS à extraire : plus on a d'informations préalable sur le type de donnée, le dimensionnement et l'éventuelle calibration d'un SDS, plus on peut utiliser une fonction efficace, ce qui, pour de très gros SDS, peut avoir une importance non négligeable.

Le fait de pouvoir dimensionner et typer un tableau rigoureusement comme le SDS à extraire permet d'utiliser la fonction la plus efficace (`hgsdsrsc` ou `hgsdsru`). Au contraire, sans ces informations, les fonctions à utiliser devront opérer d'éventuels conversions et/ou calculs d'indices pénalisants.

## Compilation

```
make (tout simplement, dans le répertoire d'installation)
```

## Utilisation

Pour utiliser la bibliothèque, il suffit de compiler le programme appelant avec les options :

```
-L<libdir> -licaf77hdf
```

où `<libdir>` représente le chemin du répertoire où se trouve cette bibliothèque.

Dans le répertoire `include` se trouve un fichier d'inclusion (`icahdf.inc`) qui contient la déclaration de toutes les fonctions disponibles, et qu'il suffit d'inclure par l'instruction :

```
include 'icahdf.inc'
```

Des exemples illustrant les différentes fonctions disponibles se trouvent dans le répertoire `examples` de l'installation.



## Note sur la calibration

Très souvent, pour limiter l'encombrement des données stockées dans les fichiers HDF, celles-ci sont en fait stockées dans un type de donnée plus petit après une opération de « calibration ».

Le terme « calibration » au sens HDF (et utilisé tout au long de ce manuel) correspond au passage des valeurs physiques aux valeurs de codage dans le fichier HDF. Pour l'opération inverse qui consiste à rétablir les valeurs physiques originales à partir des valeurs réellement stockées dans le fichier, et des coefficients de calibration, on parlera ici de « décalibration ».

La convention prise par HDF est la suivante : si *pen* et *off* sont les coefficients de calibration, et si *sds\_value* représente la valeur de codage effectivement stockée dans le SDS, la valeur physique originale *orig\_value* est restituée selon la formule :

```
orig_value = pente*(sds_value - offset)
```

C'est la même formule qui est appliquée ici. Néanmoins, ce n'est qu'une convention et un SDS peut parfaitement avoir été « calibré » d'une manière différente. C'est pourquoi chaque fonction qui effectue ce calcul se double d'une seconde fonction exécutant la même tâche, mais en appliquant une autre formule, fournie par l'utilisateur sous la forme d'une fonction externe.

Par exemple, si on veut que la formule appliquée soit plutôt, comme c'est souvent le cas :

```
orig_value = pente*sds_value + offset
```

on passera en argument une fonction du genre :

```
function mavaleur(sds_value, pente, offset)
real*8 mavaleur, sds_value, pente, offset
mavaleur = pente*sds_value + offset
end
```

qu'il faudra déclarer comme 'external' dans le programme appelant. Les coefficients de calibration HDF étant toujours de type `real*8`, tout le calcul se fait dans ce type ; une conversion ultérieure éventuelle aura lieu.

Un exemple d'utilisation d'une telle fonction est fourni dans le programme `example4.f`.



## Fonction HDFSINFO

Cette fonction récupère un certain nombre d'informations sur un fichier HDF : nombre de SDS et d'attributs du fichier, noms des SDS et noms des attributs.

Elle retourne 0 en cas de succès et -1 en cas d'erreur.

```
integer function hdfsinfo(hdf_file, nsds, natt, nsdsnam, sds_name, nattnam,  
+                          att_name)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
hdf_file	I	character*(*)	Chemin du fichier HDF
nsds	O	integer	Nombre de SDS du fichier
natt	O	integer	Nombre d'attributs du fichier
nsdsnam	I	integer	Dimension du tableau sds_name
sds_name	O	character*(*)	Tableau de dimension nsdsnam fourni par le programme appelant pour stocker les noms des SDS du fichier
nattnam	I	integer	Dimension du tableau att_name
att_name	O	character*(*)	Tableau de dimension nattnam fourni par le programme appelant pour stocker les noms des attributs du fichier

Le nombre de noms de SDS (respectivement d'attributs) retourné dans le tableau sds\_name (respectivement. att\_name) sera limité à la valeur de nsdsnam (respectivement. nattnam) ; en particulier, aucun nom ne sera retourné si celui-ci est nul.



## Fonction HSDSINFO

Cette fonction récupère un certain nombre d'informations sur un SDS d'un fichier HDF : type (calibré et/ou décalibré), dimensions, coefficients de calibration éventuels, noms des attributs.

Elle retourne 0 en cas de succès et -1 en cas d'erreur.

```
integer fonction hsdinfo(hdf_file, sds_name, nval, ctype, utype, clbsw,  
+                        clb, rank, dimsizes, natt, nattnam, att_name)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
hdf_file	I	character*(*)	Chemin du fichier HDF
nval	O	integer	Nombre de valeurs du SDS
ctype	O	integer	Type de donnée brute du SDS extrait
utype	O	integer	Type de donnée décalibrée du SDS
clbsw	O	logical	Indique si le SDS est calibré
clb	O	real*8(4)	Coefficients de calibration des données du SDS. Dans l'ordre : pente, erreur sur la pente, offset, erreur sur l'offset ; sans signification si le SDS n'est pas calibré (dans ce cas, clbsw vaut .false.)
rank	O	integer	Nombre de dimensions du SDS
dimsizes	O	integer(*)	Valeurs des dimensions du SDS. Ce tableau doit être dimensionné dans le programme appelant. Le mieux est d'inclure le fichier netcdf.f90 et de dimensionner le tableau à la valeur MAXNCDIM (nombre maximum de dimensions d'un SDS).
natt	O	integer	Nombre d'attributs du SDS
nattnam	I	integer	Dimension du tableau att_name
att_name	O	character*(*)	Tableau de dimension nattnam fourni par le programme appelant pour stocker les noms des attributs du SDS.

Le nombre de noms d'attributs retourné dans le tableau att\_name sera limité à la valeur de nattnam ; en particulier, aucun nom ne sera retourné si celui-ci est nul.



## Fonction HGSDSRC

Cette fonction est la plus simple et la plus rapide pour récupérer les données d'un SDS ; elle extrait les données du SDS de manière brute directement dans le tableau `data` fourni par le programme appelant.

Si le SDS est calibré, il n'y a pas de décalibration des données extraites (au contraire des fonctions `hgdsru` et `hgdsrus`).

Elle retourne 0 en cas de succès et -1 en cas d'erreur.

```
integer fonction hgdsrsc(hdf_file, sds_name, n, data)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
<code>hdf_file</code>	I	<code>character*(*)</code>	Chemin du fichier HDF
<code>sds_name</code>	I	<code>character*(*)</code>	Nom du SDS à extraire
<code>n</code>	O	<code>integer</code>	Taille en octets du tableau fourni par le programme appelant
<code>data</code>	O	???	Tableau fourni par le programme appelant pour stocker les données extraites du SDS.

Le type du tableau `data` fourni n'a aucune importance, la seule contrainte étant que sa taille en octets doit être suffisante pour contenir les données brutes du SDS. Par exemple, si le SDS est un tableau de dimensions (50,100) et de type `integer*2` (2 octets), et si le tableau `data` est de type `character` (1 octet), `n` devra valoir au moins  $50*100*2 = 10000$ .

Un appel préalable à la fonction `hsdsinfo` est pratiquement incontournable pour pouvoir dimensionner le tableau `data`, éventuellement le décalibrer et, bien entendu, connaître son type de donnée.



## Fonctions HGSDSRU et HGSDSRUS

Ces deux fonctions sont les plus simples et la plus rapides pour récupérer les données d'un SDS ; elles extraient les données du SDS directement dans le tableau `data` fourni par le programme appelant.

Si le SDS est calibré, il y a automatiquement décalibration des données extraites (au contraire de la fonction `hgsdsrc`), sinon, le fonctionnement est rigoureusement équivalent à celui de `hgsdsrc`.

La différence entre les deux fonctions réside en ce que la fonction `hgsdsrc` prend un argument supplémentaire `cal_func` qui est une fonction permettant de modifier la signification par défaut des paramètres de calibration (voir « Note sur la calibration »).

Elles retournent 0 en cas de succès et -1 en cas d'erreur.

```
integer function hgsdsru (hdf_file, sds_name, n, data)
```

```
integer function hgsdsrc(hdf_file, sds_name, n, data, cal_func)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
<code>hdf_file</code>	I	<code>character*(*)</code>	Chemin du fichier HDF
<code>sds_name</code>	I	<code>character*(*)</code>	Nom du SDS à extraire
<code>n</code>	O	<code>integer</code>	Taille en octets du tableau fourni par le programme appelant
<code>data</code>	O	???	Tableau fourni par le programme appelant pour stocker les données extraites du SDS.
<code>cal_func</code>	I	<code>real*8</code> <code>function</code>	Calcule la valeur originale à partir de la valeur stockée et des coefficients de calibration

Le type du tableau `data` fourni n'a aucune importance, la seule contrainte étant que sa taille en octets doit être suffisante pour contenir les données (éventuellement décalibrée) du SDS. Par exemple, si le SDS est un tableau de dimensions (50,100) dont le type de donnée décalibré (ou réel si le SDS n'est pas calibré) est `real*8` (8 octets), et si le tableau `data` est de type `integer*2` (2 octets), alors `n` devra valoir au moins  $50 \times 100 \times 8 / 2 = 20000$ .

Ici encore, un appel préalable à la fonction `hdsinfo` est pratiquement incontournable pour pouvoir dimensionner le tableau `data` et connaître son type de donnée.



## Fonction HGSDSUC

Cette fonction permet de récupérer les données brutes d'un SDS dans un type préalablement fixé (pas forcément le type de donnée réel du SDS, d'ailleurs) ; il y a donc éventuellement conversion de type.

Si le SDS est calibré, il n'y a pas de décalibration des données extraites (au contraire des fonctions `hgdsu` et `hgdsuus`).

Elle retourne 0 en cas de succès et -1 en cas d'erreur.

```
integer function hgdsuc(hdf_file, sds_name, n, data, dtype, nval, ctype,  
+                       utype, clbsw, clb, rank, dimsizes)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
<code>hdf_file</code>	I	<code>character*(*)</code>	Chemin du fichier HDF
<code>sds_name</code>	I	<code>character*(*)</code>	Nom du SDS à extraire
<code>n</code>	I	<code>integer</code>	Taille en octets du tableau fourni par le programme appelant
<code>data</code>	O	???	Tableau fourni par le programme appelant pour stocker les données extraites du SDS
<code>dtype</code>	I	<code>integer</code>	Type de donnée du tableau fourni
<code>nval</code>	O	<code>integer</code>	Nombre de valeurs du SDS
<code>ctype</code>	O	<code>integer</code>	Type de donnée brute du SDS extrait
<code>utype</code>	O	<code>integer</code>	Type de donnée décalibrée du SDS
<code>clbsw</code>	O	<code>logical</code>	Indique si le SDS est calibré
<code>clb</code>	O	<code>real*8(4)</code>	Coefficients de calibration des données du SDS. Dans l'ordre : pente, erreur sur la pente, offset, erreur sur l'offset ; sans signification si le SDS n'est pas calibré ( <code>clbsw</code> vaut <code>.false.</code> ).
<code>rank</code>	O	<code>integer</code>	Nombre de dimensions du SDS
<code>dimsizes</code>	O	<code>integer(*)</code>	Valeurs des dimensions du SDS. Ce tableau doit être dimensionné dans le programme appelant. Le mieux est d'inclure le fichier <code>netcdf.f90</code> et de dimensionner le tableau à la valeur <code>MAXNCDIM</code> (nombre maximum de dimensions d'un SDS)

C'est le paramètre `dtype` qui influe ici sur le fonctionnement de la fonction. Si ce paramètre est initialisé avec la valeur d'un type HDF valide (cf. documentation HDF pour FORTRAN), les données extraites du SDS seront retournées dans le tableau `data` après une éventuelle conversion dans ce type (on aura tout intérêt à inclure le fichier `hdf.f90` contenant les valeurs des types valides sous forme de constantes).

Le type du tableau `data` fourni n'a, ici encore, pas réellement d'importance, la seule contrainte étant toujours que sa taille en octets soit suffisante pour contenir les données (éventuellement converties) du SDS.

Par exemple, si le SDS est un tableau de dimensions (50,100), d'un type de donnée brute quelconque, et si le paramètre `dtype` correspond au type `real*8` (8 octets), c'est en fait un tableau de dimensions (50,100) de `real*8` qu'il faut pouvoir stocker. Rien n'empêche alors que le tableau `data` soit en fait de type `byte`, mais n devra alors valoir au moins  $50 \times 100 \times 8 = 40000$ .



Néanmoins, il est évident qu'il sera plus aisé d'utiliser un tableau `data` dont le type correspond à l'argument `dtype`.

Si `dtype` contient une valeur invalide pour un type HDF, la fonction ne peut pas effectuer de conversion et se comporte alors exactement comme `hgsdsrc`. Son appel peut alors remplacer un appel à `hdsinfo` suivi d'un appel à `hgsdsrc`.



## Fonctions HGSDSUU et HGSDSUUS

Ces fonctions permettent de récupérer les données éventuellement décalibrées d'un SDS dans un type préalablement fixé (pas forcément le type de donnée réel du SDS, d'ailleurs); il y a donc éventuellement conversion de type.

Si le SDS est calibré, il y a automatiquement décalibration des données extraites (au contraire de la fonction `hgdsuc`), sinon, le fonctionnement est rigoureusement équivalent à celui de `hgdsuc`, à ceci près que les arguments `ctype`, `clbsw` et `clb`, de toute façon inutiles, n'existent pas.

La différence entre les deux fonctions réside en ce que la fonction `hgdsuus` prend un argument supplémentaire `cal_func` qui est une fonction permettant de modifier la signification par défaut des paramètres de calibration (voir « Note sur la calibration »).

Elles retournent 0 en cas de succès et -1 en cas d'erreur.

```
integer function hgdsuu (hdf_file, sds_name, n, data, dtype, nval, utype,  
+                        rank, dimsizes)
```

```
integer function hgdsuus(hdf_file, sds_name, n, data, dtype, nval, utype,  
+                        rank, dimsizes, cal_func)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
<code>hdf_file</code>	I	<code>character*(*)</code>	Chemin du fichier HDF
<code>sds_name</code>	I	<code>character*(*)</code>	Nom du SDS à extraire
<code>n</code>	I	<code>integer</code>	Taille en octets du tableau fourni par le programme appelant
<code>data</code>	O	???	Tableau fourni par le programme appelant pour stocker les données extraites du SDS
<code>dtype</code>	I	<code>integer</code>	Type de donnée du tableau fourni
<code>nval</code>	O	<code>integer</code>	Nombre de valeurs du SDS
<code>utype</code>	O	<code>integer</code>	Type de donnée décalibrée du SDS
<code>rank</code>	O	<code>integer</code>	Nombre de dimensions du SDS
<code>dimsizes</code>	O	<code>integer(*)</code>	Valeurs des dimensions du SDS. Ce tableau doit être dimensionné dans le programme appelant. Le mieux est d'inclure le fichier <code>netcdf.f90</code> et de dimensionner le tableau à la valeur <code>MAXNCDIM</code> (nombre maximum de dimensions d'un SDS).
<code>cal_func</code>	I	<code>real*8</code> <code>function</code>	Calcule la valeur originale à partir de la valeur stockée et des coefficients de calibration

C'est le paramètre `dtype` qui influe ici sur le fonctionnement de la fonction. Si ce paramètre est initialisé avec la valeur d'un type HDF valide (cf. documentation HDF pour FORTRAN), les données extraites du SDS seront retournées dans le tableau `data` après une éventuelle décalibration puis une éventuelle conversion dans ce type (on aura tout intérêt à inclure le fichier `hdf.f90` contenant les valeurs des types valides sous forme de constantes).

Le type du tableau `data` fourni n'a, ici encore, pas réellement d'importance, la seule contrainte étant toujours que sa taille en octets soit suffisante pour contenir les données (éventuellement décalibrées et/ou converties) du SDS.



Par exemple, si le SDS est un tableau de dimensions ( 50 , 100 ), d'un type de donnée décalibrée quelconque, et si le paramètre `dtype` correspond au type `real*8` (8 octets), c'est en fait un tableau de dimensions ( 50 100 ) de `real*8` qu'il faut pouvoir stocker. Rien n'empêche alors que le tableau `data` soit en fait de type `byte`, mais n devra alors valoir au moins  $50*100*8 = 40000$ .

Néanmoins, ici aussi il est évident qu'il sera plus aisé d'utiliser un tableau `data` dont le type correspond à l'argument `dtype`.

Si `dtype` contient une valeur invalide pour un type HDF, la fonction ne peut pas effectuer de conversion et se comporte alors exactement comme `hgsdsru`. Son appel peut alors remplacer un appel à `hgsdsinfo` suivi d'un appel à `hgsdsru`, à ceci près que les arguments `ctype`, `clbsw` et `clb`, de toute façon inutiles puisque les données extraites sont décalibrées, n'existent pas.



## Fonction HGSDSDC

Cette fonction permet de récupérer les données brutes d'un SDS dans un type et un dimensionnement préalablement fixés (pas forcément ceux du SDS) ; il y a donc éventuellement conversion de type et déplacement dans les tableaux.

Si le SDS est calibré, il n'y a pas de décalibration des données extraites (au contraire des fonctions `hgdsdu` et `hgdsdus`).

Elle retourne 0 en cas de succès et -1 en cas d'erreur.

```
integer function hgdsdc(hdf_file, sds_name, data, dtype, nd, dims, nval,  
+                       ctype, utype, clbsw, clb, rank, dimsizes)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
<code>hdf_file</code>	I	<code>character*(*)</code>	Chemin du fichier HDF
<code>sds_name</code>	I	<code>character*(*)</code>	Nom du SDS à extraire
<code>data</code>	O	???	Tableau fourni par le programme appelant pour stocker les données extraites du SDS.
<code>dtype</code>	I	<code>integer</code>	Type de donnée du tableau fourni
<code>nd</code>	I	<code>integer</code>	Nombre de dimensions du tableau fourni
<code>dims</code>	I	<code>integer(*)</code>	Tableau des valeurs des dimensions du tableau fourni
<code>nval</code>	O	<code>integer</code>	Nombre de valeurs du SDS
<code>ctype</code>	O	<code>integer</code>	Type de donnée brute du SDS extrait
<code>utype</code>	O	<code>integer</code>	Type de donnée décalibrée du SDS
<code>clbsw</code>	O	<code>logical</code>	Indique si le SDS est calibré
<code>clb</code>	O	<code>real*8(4)</code>	Coefficients de calibration des données du SDS. Dans l'ordre : pente, erreur sur la pente, offset, erreur sur l'offset ; sans signification si le SDS n'est pas calibré ( <code>clbsw</code> vaut <code>.false.</code> )
<code>rank</code>	O	<code>integer</code>	Nombre de dimensions du SDS
<code>dimsizes</code>	O	<code>integer(*)</code>	Valeurs des dimensions du SDS. Ce tableau doit être dimensionné dans le programme appelant. Le mieux est d'inclure le fichier <code>netcdf.f90</code> et de dimensionner le tableau à la valeur <code>MAXNCDIM</code> (nombre maximum de dimensions d'un SDS)

Le but est ici de recueillir les données brutes extraites du SDS avec la même structure dimensionnelle, la contrainte étant que le tableau fourni ait un nombre de dimensions au moins égal à celui du SDS et que chacune de ses dimensions soit au moins égale à la dimension correspondante du SDS. Par exemple, si le SDS est de dimensions (50, 20, 10), un tableau de dimension (100, 20, 20, 5) sera adapté, mais pas un tableau de dimensions (500, 500) ou (10, 100, 10), même si le nombre d'éléments est suffisant.



De plus, cette fois, le paramètre `dtype` devra cette fois obligatoirement correspondre au type de donnée du tableau `data` (ou au moins à un type de même taille), sinon le tableau obtenu sera inexploitable. Si ce paramètre est initialisé avec la valeur d'un type HDF valide (cf. documentation HDF pour FORTRAN), les données extraites du SDS seront retournées, après une éventuelle conversion dans ce type, dans le tableau `data` avec les mêmes indices dimensionnels que ceux qu'ils avaient dans le SDS (on aura tout intérêt à inclure le fichier `hdf.f90` contenant les valeurs des types valides sous forme de constantes).

Si `dtype` contient une valeur invalide pour un type HDF, la fonction ne peut pas effectuer de conversion et considère alors que le tableau `data` est du même type que les données à stocker, ce qui peut provoquer des résultats imprévisibles si tel n'est pas le cas.



## Fonctions HGSDSDU et HGSDSDUS

Ces fonctions permettent de récupérer les données éventuellement décalibrées d'un SDS dans un type et un dimensionnement préalablement fixés (pas forcément ceux du SDS) ; il y a donc éventuellement conversion de type et déplacement dans les tableaux.

Si le SDS est calibré, il y a automatiquement décalibration des données extraites (au contraire de la fonction `hgsdsdc`), sinon, le fonctionnement est rigoureusement équivalent à celui de `hgsdsdc`, à ceci près que les arguments `ctype`, `clbsw` et `clb`, de toute façon inutiles, n'existent pas.

La différence entre les deux fonctions réside en ce que la fonction `hgsdsdus` prend un argument supplémentaire `cal_func` qui est une fonction permettant de modifier la signification par défaut des paramètres de calibration (voir « Note sur la calibration »).

Elles retournent 0 en cas de succès et -1 en cas d'erreur.

```
integer function hgsdsdu (hdf_file, sds_name, data, dtype, nd, dims, nval,  
+                          utype, rank, dimsizes)
```

```
integer function hgsdsdus(hdf_file, sds_name, data, dtype, nd, dims, nval,  
+                          utype, rank, dimsizes, cal_func)
```

ARGUMENT	I/O	TYPE	DESCRIPTION
<code>hdf_file</code>	I	<code>character*(*)</code>	Chemin du fichier HDF
<code>sds_name</code>	I	<code>character*(*)</code>	Nom du SDS à extraire
<code>data</code>	O	???	Tableau fourni par le programme appelant pour stocker les données extraites du SDS.
<code>dtype</code>	I	<code>integer</code>	Type de donnée du tableau fourni
<code>nd</code>	I	<code>integer</code>	Nombre de dimensions du tableau fourni
<code>dims</code>	I	<code>integer(*)</code>	Tableau des valeurs des dimensions du tableau fourni
<code>nval</code>	O	<code>integer</code>	Nombre de valeurs du SDS
<code>utype</code>	O	<code>integer</code>	Type de donnée décalibrée du SDS
<code>rank</code>	O	<code>integer</code>	Nombre de dimensions du SDS
<code>dimsizes</code>	O	<code>integer(*)</code>	Valeurs des dimensions du SDS. Ce tableau doit être dimensionné dans le programme appelant. Le mieux est d'inclure le fichier <code>netcdf.f90</code> et de dimensionner le tableau à la valeur <code>MAXNCDIM</code> (nombre maximum de dimensions d'un SDS).
<code>cal_func</code>	I	<code>real*8</code> <code>function</code>	Calcule la valeur originale à partir de la valeur stockée et des coefficients de calibration

Le but est ici de recueillir les données extraites du SDS, éventuellement décalibrées, avec la même structure dimensionnelle, la contrainte étant que le tableau fourni ait un nombre de dimensions au moins égal à celui du SDS et que chacune de ses dimensions soit au moins égale à la dimension correspondante du SDS. Par exemple, si le SDS est de dimensions (50,20,10), un tableau de dimension (100,20,20,5) sera adapté, mais pas un tableau de dimensions (500,500) ou (10,100,10), même si le nombre d'éléments est suffisant.



De plus, cette fois, le paramètre `dtype` devra cette fois obligatoirement correspondre au type de donnée du tableau `data` (ou au moins à un type de même taille), sinon le tableau obtenu sera inexploitable. Si ce paramètre est initialisé avec la valeur d'un type HDF valide (cf. documentation HDF pour FORTRAN), les données extraites du SDS seront retournées, après une éventuelle décalibration et une éventuelle conversion dans ce type, dans le tableau `data` avec les mêmes indices dimensionnels que ceux qu'ils avaient dans le SDS (on aura tout intérêt à inclure le fichier `hdf.f90` contenant les valeurs des types valides sous forme de constantes).

Si `dtype` contient une valeur invalide pour un type HDF, la fonction ne peut pas effectuer de conversion et considère alors que le tableau `data` est du même type que les données à stocker, ce qui peut provoquer des résultats imprévisibles si tel n'est pas le cas.