Fujitsu Software BS2000 C/C++
Version 4.0C
June 2026

Release Notice

1 General

C/C++ V4.0 is the follow-up version of C/C++ V3.2.

The name of the delivery group and a component part of the product file names is CPP.

C/C++ is the strategic BS2000 compiler for developing and porting applications from the open world (e.g. OO applications) to BS2000 Business Servers.

*10 The release level is that of: January 2026.

*1 Changes to release level December 2019 are marked with “*1”.

*2 Changes to release level November 2020 are marked with “*2”.

*3 Changes to release level November 2021 are marked with “*3”.

*4 Changes to release level June 2022 are marked with “*4”.

*5 Changes to release level June 2023 are marked with “*5”.

*6 Changes to release level November 2023 are marked with “*6”.

*7 Changes to release level June 2024 are marked with “*7”.

*8 Changes to release level November 2024 are marked with “*8”.

*9 Changes to release level June 2025 are marked with “*9”.

*10 Changes to release level November 2025 are marked with “*10”.

*3 This and other Release Notice(s) are available online at
*3 https://bs2manuals.ts.fujitsu.com/.

If one or more previous versions are skipped when this product version is used, the information from the Release Notices of the previous versions must be noted.

1.1 Ordering

C/C++ V4.0 can be ordered from your local distributors.

1.2 Delivery

The C/C++ V4.0 files are supplied via SOLIS.

The following files are part of the delivery scope of C/C++ V4.0:

	SINLIB.CPP.040	Compiler library (POSIX)
	SINPRC.CPP.040	Library with private installation procedures
*2	(deleted)	
	SYSLNK.CPP.040	Compiler library (BS2000)
	SYSMES.CPP.040	Message file
	SYSSDF.CPP.040	SDF syntax file
	SYSSDF.CPP.040.IU.USER	SDF user syntax file for II-UPDATE
	SYSSDF.CPP.040.USER	SDF user syntax file for private installation
	SYSSII.CPP.040	IMON installation file
	SYSSPR.CPP.040.IU	SDF procedure for START-II-UPDATE
	SYSSSC.CPP.040.POSIX	Subsystem declaration (POSIX)
	SYSSSC.CPP.040	Subsystem declaration (BS2000)

The current file and volume characteristics are listed in the SOLIS2 delivery cover letter.

1.3 Documentation

The following descriptions are available for C/C++ V4.0:

German version	English version

C language description:

Programmieren in C 2. Ausgabe ANSI-C Kernighan und Ritchie	The C Programming Language 2nd Edition - ANSI-C Kernighan and Ritchie

C++ language description:

Die C++-Programmiersprache 2. Ausgabe von Bjarne Stroustrup	The C++ Programming Language (2nd Edition) by Bjarne Stroustrup
Die C++-Programmiersprache 3. Ausgabe von Bjarne Stroustrup	The C++ Programming Language (3rd Edition) by Bjarne Stroustrup

Compiler manual (general part and SDF syntax):

C/C++ V4.0 Benutzerhandbuch (BHB)	C/C++ V4.0 User Guide

Compiler manual (POSIX Syntax):

C/C++ V4.0 POSIX-Kommandos Benutzerhandbuch	C/C++ V4.0 POSIX Commands User Guide

C library functions:

C-Bibliotheksfunktionen Referenzhandbuch	C Library Functions Reference Manual

C library functions for POSIX:

C-Bibliotheksfunktionen für POSIX-Anwendungen Referenzhandbuch	C Library Functions for POSIX Applications Reference Manual

C++ library functions for V3 mode (ANSI-C++):

-	Standard C++ Library V1.2 User's Guide and Reference

C++ library functions for V2 mode (Cfront):

C++ V2.1 C++-Bibliotheksfunktionen	C++ V2.1 C++ Library Functions

Tools.h++ class library (for V3 mode, ANSI-C++):

-	Tools.h++ V7.0 User's Guide
-	Tools.h++ V7.0 Class Reference

The following additional documentation is also available that is not intended exclusively for C/C++ users:

CRTE runtime system:

*8	as of	CRTE V21.0 Common Runtime Environment Benutzerhandbuch	CRTE V21.0 Common Runtime Environment User Guide

The manuals of the BS2000 basic configuration are additionally required for operating C/C++.

*3 The documentation is available on the Internet at
*3 https://bs2manuals.ts.fujitsu.com. There you will find both individual manuals and
*3 (under the "Softbooks" tab) the ISO image of a DVD with the entire inventory.

2 Software extensions

Only the extensions and improvements over the previous version C/C++ V3.2 are described in the following section.

2.1 Corrections of known errors

C/C++ V4.0 contains a number of error corrections which should be used.

2.2 Support of the C11 standard

The compiler accepts code according to the C11 standard.
*1 There are some optional features from C11 which CPP 4.0 does not support.
Any restrictions are described in the manuals to C/C++ V4.0.

2.3 Support of the C++2017 standard

The compiler accepts code according to the C++2017 standard.
*1 There are a few mandatory features from C++2017 which CPP 4.0 does not sup-
*1 port.
Any restrictions are described in the manuals to C/C++ V4.0.

*1 2.4 #pragma instantiate_inline

*1 The compiler supports the `#pragma instantiate_inline`.

*2 2.5 Support of the C++2020 standard

*2 The compiler accepts code according to the C++2020 standard.
*2 There are a few mandatory features from C++2020 which CPP 4.0 does not sup-
*2 port. The library of the C++2020 standard is not supported, but the library of the
*2 C++2017 standard can be used in this mode.
*2 Any restrictions are described in the manuals to C/C++ V4.0.

*2 2.6 Format check for printf and scanf functions

*2 The format check for the functions fprintf(), printf(), sprintf(), snprintf(), fscanf(),
*2 scanf() and sscanf() as well as their ASCII- and IEEE-version is activated (for
*2 more information see user manual pragmas `__printf_args` bzw.
`__scanf_args`).
*2 This format check requires CRTE V11.1A40 or V21.0A10.

*3 2.7 C++ predefines

*3 The C++ predefines (starting with „__cpp“) are defined. These indicate the pres-
*3 ence of certain C++ features in the compiler.
*3 Some of the affected features depend on the language mode. Accordingly, the
*3 value of the corresponding predefine depends on the language mode.

*3 2.8 C++2020 „using enum“ feature

*3 The statement „using enum E;“ is supported. Thus the enumerators of „enum
*3 class E{…};“ can be addressed better.

*4 2.9 Support for library versions

*5 There is a new command //MODIFY-LIBRARY-VERSION to specify the C++ li-
*5 brary versions.
*4 Further details of this command are described in the C/C++ V4.0 manuals.

*7 2.10 Support of %m in format strings of the printf family

*7 Specifying `printf("%m")` is equivalent to specifying `printf ("%s", strerror(errno));`

*7 2.11 Bitfield with type “long long”

*7 The compiler accepts bitfield with type “long long”, from this version onwards.

3 Technical information

3.1 Resource requirements

The following memory range is required in the system address space for running C/C++:

at least 64 MB class 6 memory

This value represents the minimum requirement that may increase due to the amount of data involved and the application (e.g. when using templates in C++ sources).

*8 C/C++ V4.0 occupied after the load 4735 pages in class 6 memory, with preloaded
*8 subsystem CPP 2202 pages.

The memory required for preloading the CPP subsystem is approximately 5,4 MB. The memory required for preloading the CPPP subsystem is approximately 5,4 MB.

3.2 Software configuration

*8 BS2000 BS2000 OS DX V1.0 is required for C/C++ V4.0.

C/C++ V4.0 requires the following correction levels of software products:

*8
- CRTE-BASYS V21.0A
- LLMAM as of V3.4A30

and the products: BINDER, BUILDER, CRTE, PLAM and SDF used in releases matching to the OSD version.

Additional software used in releases matching to the OSD version is required for using particular functions:

- BLSSERV for dynamic binding/loading
- DSSM for preloading the compiler
- LMS for private C/C++ installations
- POSIX-HEADER for using the compiler in POSIX
- POSIX-BC for using the compiler in POSIX

3.3 Product installation

3.3.1 Installation and deinstallation

C/C++ V4.0 consists of the components (compiler, listing generator, II-update tool) with an SDF interface that can be used in the BS2000, and the POSIX components (compiler, listing generator, tools) which can be optionally installed for use under POSIX.

C/C++ V4.0 is normally installed in the BS2000 using SOLIS or IMON. If required, the POSIX parts of C/C++ must be installed manually, unless IMON as of version 2.8 is used. With IMON as of version 2.8 the POSIX parts of the product can already be automatically installed by SOLIS.

All in all support is provided for the following installation types, which are described in more detail below:

- Public installation (SOLIS) for BS2000
- Automatic public installation (SOLIS) for POSIX
- Manual public installation for POSIX with and without IMON
- Private installation (scripts from SINPRC) for BS2000
- Private installation (scripts from SINPRC) for POSIX

A public installation is normally available for all users of a system and requires appropriate administrator privileges for the installation, whereas a private installation is mostly only intended for the user performing the installation and does not require administrator privileges.

No update installation is supported. The old version must be deinstalled before a new version or a correction version is installed. This is also valid in case errors occur during the installation.

The deinstallation of an older correction version (not an older version) is also possible using a newer SINLIB of the same release unit and must therefore no longer be performed before the SOLIS volume is installed.

If you want to install several versions and/or correction versions in parallel, you must ensure that the product files for each installation are available separately. This means that, e.g. for versions in which only the correction version differs, they must be located under different user ids or that the name of the release items must be different (e.g. prefix). The installation directory for the POSIX part of the installation must be selected differently from every other installation.

3.3.1.1 Public installation (SOLIS) for BS2000

This is the standard installation variant for the BS2000 part of the product and should be the recommended installation variant for most customers. In this regard, SOLIS automatically takes over all the tasks, such as the placing of the product files (in accordance with the installer), activation of syntax and message files and registration of the preloadable subsystems. The possibilities offered by SOLIS for the installation on any user ids and/or with modified file names are supported, as are parallel installations under different product versions.

3.3.1.1.1 Performing the installation

See the SOLIS/IMON documentation for the description of the SOLIS installation.

3.3.1.1.2 Preloadable subsystems

The compiler for the BS2000 and also the compiler for POSIX are available as preloadable subsystems. The relevant subsystem declarations have already been entered in the system catalog for the SOLIS installation. The subsystem names are: CPP and CPPP.

The respective system administration must decide which one of the subsystems is preloaded and whether this already takes place during the system start. However, preloading considerably reduces the load times when the compiler is called.

All of the above-named subsystems can theoretically be preloaded in parallel. On the other hand, parallel preloading of subsystems of the same name but of a different version is not permitted.

3.3.1.1.3 Use

After successful installation with SOLIS, the compiler and the listing generator can be called via their start commands without any further action. On the other hand, an additional manual installation step, which is described in section 3.3.1.7, is almost always required to use the II-Update tool.

However, you should ensure that only the product version or variant that was last installed publicly in the BS2000 is and can be the owner of the start commands START-CPLUS-COMPILER and START-CPLUS-LISTING-GENERATOR.

It is therefore not recommended for several product variants or versions to be installed publically in parallel, but it is possible in principle. The other installations can then only be called using the START-PROGRAM or the START-EXECUTABLE-PROGRAM, e.g. with:

```
/START-EXE *LIB($.SYSLNK.CPP.040,SDFCC)
```

or

```
/START-EXE *LIB($.SYSLNK.CPP.040,SDFLISTGEN)
```

3.3.1.1.4 Deinstallation

Deinstallation of the product is also performed using SOLIS. However, it should be noted that a manually installed POSIX part of the product must also be deinstalled manually beforehand.

3.3.1.2 Automatic public installation (SOLIS) for POSIX

This installation variant is possible on systems with IMON as of V2.8 and should be the recommended installation variant there for most customers. The POSIX part of the installation takes place automatically during or after the normal SOLIS/IMON installation of the product and no further manual installation step is required.

This installation variant is simple and convenient, but has certain restrictions:

1. The installation path cannot be freely selected. The product is always installed in POSIX under the standard path /opt/C.

2. Before the installation all versions of the same release unit that have already been publically installed in POSIX are deinstalled, regardless of the version and the installation location. This installation variant does not permit any public parallel installations of different versions and/or correction versions.

3.3.1.2.1 Performing the installation

See the SOLIS/IMON documentation for the description of the SOLIS installation. See the documentation for IMON V2.8 for how IMON is induced to perform the automatic POSIX installation.

3.3.1.2.2 Preloadable subsystem

A publically installed POSIX compiler can be preloaded as the subsystem by the system administrator (see section 3.3.1.1.2), which reduces the load times when starting the compiler. On the other hand, the compiler cannot be preloaded with the POSIX loader posdbl.

3.3.1.2.3 Use

After the installation has been completed, the compiler and the listing generator can be called using their POSIX commands (e.g. cc) without any further action.

3.3.1.2.4 Deinstallation

Deinstallation of the POSIX part also takes place automatically if another product version of C/C++ is installed in this way or if the delivery unit is removed with SOLIS/IMON from the system.

3.3.1.3 Manual public installation for POSIX with IMON

The installation is performed in accordance with the POSIX Basic Principles Manual, in the section entitled Delivery and installation procedure for POSIX program packages. This is the recommended installation variant for all systems with an IMON version older than V2.8.

It is possible to choose the installation path in the POSIX file system, but it should not be already occupied by other products or installations. As a result of selecting an installation path the public parallel installation of different versions and/or correction versions of C/C++ is possible.

3.3.1.3.1 Prerequisites

The installation must be performed on a privileged basis under the system default userid (mostly TSOS) and the release items must be installed with IMON, i.e. also be registered. Ideally, this installation requires the prior public installation for BS2000.

3.3.1.3.2 Performing the installation

The POSIX installation is started with

```
/START-POSIX-INSTALLATION
```

In the following mask it is necessary to select "Install packages on POSIX".
The following values can be entered in the next mask:

IMON:	Y
Product:	CPP
Package:	
Version:	V04.0 (optional)
Correction version:	A00 (or a different version, optional)

Specification of the version and/or correction version is only required if several versions of the product are registered with IMON.

After the screen has been sent, the mask is shown again, and a field to enter the required installation directory is now also shown. The field has the standard installation path default /opt/C.

The field can now be modified. This can make sense if you want to install several versions and/or correction versions of a product in parallel. The installation scripts check whether another product is already installed under this path, in which case the installation is aborted.
The installation starts automatically as soon as the screen is sent.

3.3.1.3.3 Errors during the installation

As the output of installation scripts is not always displayed correctly by the POSIX installation tool, a file is stored under /var/tmp/inst.<release_unit> in the case of an error, from which more exact information about the cause of the error can possibly be provided. The POSIX installation tool then shows an error message, which refers to a problem during the execution of the product-specific scripts, e.g.:

shell script ".../.install_pre" reports error 102.

However, the POSIX installation tool does not always abort the installation when errors occur and the user must therefore by all means deinstall the incorrectly installed product before attempting another installation. In particular, a partially created installation directory is not automatically deleted again if an error occurs.
The number specified in the error message provides a reference to the error that has occurred, the actual error messages of the shell or other commands are found in the error output file in /var/tmp.

Error number	**Description**
100	The POSIX installation tool used does not have the version required for the installation of C/C++ V4.0. An update of POSIX to the most current version is necessary.
101	The installation directory was not created by the POSIX installation tool (follow-up error).
102	The specified installation path is already being used by another product installation.
103	A POSIX installation that uses the same SINLIB already exists.

3.3.1.3.4 Installation directory

The installation in the POSIX file system is into the selected directory. Symbolic links are also created for the commands to /usr/bin if and only if the standard installation path /opt/C was used for this installation.

3.3.1.3.5 Preloadable subsystem

A publically installed POSIX compiler can be preloaded as the subsystem by the system administrator (see section 3.3.1.1.2), which reduces the load times when starting the compiler. On the other hand, the compiler cannot be preloaded with the POSIX loader posdbl.

3.3.1.3.6 Use

The C/C++ commands are accessible via /usr/bin if the installation was into the standard path. This directory is entered in the standard search path of every POSIX user, and should therefore not require any further provisions for this type of installation in order to use C/C++. The commands of other C/C++ installations can be achieved by specifying the explicit command paths for the call or by entering the respective path <posix_install_path>/bin in the command search path of the caller.

3.3.1.3.7 Deinstallation

Deinstallation of the POSIX part of a public installation is performed using the POSIX installation command as TSOS by calling:

/START-POSIX-INSTALLATION

In the following mask it is necessary to select "Delete packages from POSIX". In the next mask the required installation must be selected and marked on the basis of the version and/or the installation directory. Once sent (DUE), it is necessary to confirm again with DUE.

The product can then (and not before) be removed from the system with SOLIS/IMON.

3.3.1.4 Manual public installation for POSIX without IMON

The installation is performed in accordance with the POSIX Basic Principles Manual, in the section entitled Delivery and installation procedure for POSIX program packages.

It is possible to choose the installation path in the POSIX file system, but it should not be already occupied by other products or installations. As a result of selecting an installation path the public parallel installation of different versions and/or correction versions of C/C++ is possible.

When installing without IMON, it is possible to install the same correction version of C/C++ in public repeatedly (e.g. when configuring data center userids), which is not possible with IMON.

3.3.1.4.1 Prerequisites

The installation must be performed on a privileged basis under the system default userid (mostly TSOS) and the release items may be installed on any userid.

3.3.1.4.2 Performing the installation

The POSIX installation program is started with

/START-POSIX-INSTALLATION

In the following mask it is necessary to select "Install packages on POSIX".
The following values can be entered in the next mask:

IMON:	N
Product:	CPP
Package:	
Version:	040
Userid:	*Name of the userid with the release items*

After the screen has been sent, the mask is shown again, and a field to enter the required installation directory is now also shown. The field has the standard installation path default /opt/C.

The field can now be modified. This can make sense if you want to install several versions and/or correction versions of a product in parallel. The installation scripts check whether another product is already installed under this path, in which case the installation is aborted.

The installation starts automatically as soon as the screen is sent.

3.3.1.4.3 Errors during installation

See section 3.3.1.3.3.

3.3.1.4.4 Installation directory

The installation in the POSIX file system is into the selected directory. Symbolic links are also created for the commands to /usr/bin if and only if the standard installation path /opt/C was used for this installation.

3.3.1.4.5 Preloadable subsystem

A publically installed POSIX compiler can be preloaded as the subsystem by the system administrator (see section 3.3.1.1.2), which reduces the load times when starting the compiler. On the other hand, the compiler cannot be preloaded with the POSIX loader posdbl.

The subsystem declarations are not automatically entered in the system catalog for an installation that has not been performed with SOLIS/IMON. This would have to be done manually, but is not recommended for this installation type.

3.3.1.4.6 Use

The C/C++ commands are accessible via /usr/bin if the installation was into the standard path. This directory is entered in the standard search path of every POSIX user, and should therefore not require any further provisions for this type of installation in order to use C/C++. The commands of other C/C++ installations can be achieved by specifying the explicit command paths for the call or by entering the respective path <posix_install_path>/bin in the command search path of the caller.

3.3.1.4.7 Deinstallation

The deinstallation of the POSIX part of a public installation takes place with the POSIX installation command as TSOS by calling:

```
/START-POSIX-INSTALLATION
```

In the following mask it is necessary to select "Delete packages from POSIX". In the next mask the required installation must be selected and marked on the basis of the version and/or the installation directory. Once sent (DUE), it is necessary to confirm again with DUE.

The product can then (but not before) be removed from the system with SOLIS/IMON.

3.3.1.5 Private installation for BS2000

The installation is performed using procedures that are supplied with the C/C++ distribution.

3.3.1.5.1 Prerequisites

The installation can take place under any (non-privileged) userid.

The release items of C/C++ must be accessible on this or another userid for the caller, the caller must above all also have write authorization for the file SYSLNK. The content of this file must also be in its original status and may not already have been modified by a prior private installation.

The release items that are absolutely necessary for a private installation are as follows (specified here with their original names):

```
SINPRC.CPP.040
SYSLNK.CPP.040
SYSMES.CPP.040
SYSSDF.CPP.040.USER
```

3.3.1.5.2 Performing the installation

If the product files exist under their original names on the caller's userid, the installation can then be performed with the command

```
/CALL-PROCEDURE (SINPRC.CPP.040,INSTALL.SDF)
```

In principle, the release items can be randomly renamed if this is taken into consideration during subsequent use. If the product files exist under a different userid and/or under a different name, the installation must take place in the following general form:

```
/CALL-PROCEDURE (<sinprc_name>,INSTALL.SDF), -
/   (SYSLNK=<syslnk_name>,SYSMES=<sysmes_name>, -
/   SINPRC=<sinprc_name>)
```

The names of the release items can be normal file names with/without an userid and with/without a pubset specification.

3.3.1.5.3 Errors during the installation

If the installation is ended with an error, it is possible for the file SYSLNK to already have been modified. It must be put back into the original status before another attempt at installation is made.

3.3.1.5.4 Use

After the installation has been completed, the compiler and listing generator can be used. It should be noted that the message catalog and the syntax file of the private installation must be activated at least once per session. Then the compiler or the listing generator can be called with START-PROGRAM or START-EXECUTABLE-PROGRAM.

For a private installation on the caller ID with the original names a compiler invocation could be done as follows:

```
/MOD-SDF-OPT *ADD(SYSSDF.CPP.040.USER)
/MOD-MSG-FILE ADD=SYSMES.CPP.040
/START-PROG *M(SYSLNK.CPP.040,SDFCC, -
/    R-M=A(SHARE-SCOPE=*NONE),P-M=A)
```

If the product files are available under a different userid and/or under a different name, the call of the listing generator can e.g. take place in the following general form:

```
/MOD-SDF-OPT *ADD(<sdfuser_name>)
/MOD-MSG-FILE ADD=<sysmes_name>
/START-PROG *M(<syslnk_name>,SDFLISTGEN, -
/    R-M=A(SHARE-SCOPE=*NONE),P-M=A)
```

After the start, the compiler reports with the prompt for SDF statements, analog to the start commands.

The option SHARE-SCOPE=*NONE is necessary to ensure that a link is not made to a preloaded subsystem of a different public installation with a suitable version, but possibly with a different correction version.

3.3.1.5.5 Deinstallation

Deinstallation of a private installation is done by simply deleting the release items. If the same release items also form the basis of a POSIX installation, the product must first be deinstalled in POSIX before the release items may be deleted.

3.3.1.6 Private installation for POSIX

The installation is performed using procedures that are supplied with the C/C++ distribution.

3.3.1.6.1 Prerequisites

The installation can take place under any (non-privileged) userid. This userid must be entered as a POSIX user and the POSIX file system selected for the installation must have adequate free space.

The release items of C/C++ must be accessible on this or another userid for the caller, the caller must above all also have write authorization for the file SINLIB. The content of this file must also be in its original status and may not already have been modified by a prior private installation.

The selected installation directory in the POSIX file system should not yet exist, but the user performing the installation must have the authorization to create it.

The release items that are absolutely necessary for a private installation in POSIX are as follows (specified here with their original names):

```
SINPRC.CPP.040
SINLIB.CPP.040
SYSMES.CPP.040
```

3.3.1.6.2 Performing the installation

If the product files exist under their original names on the caller's userid, the installation can then be performed with the command:

```
/CALL-PROCEDURE -
/   (SINPRC.CPP.040,INSTALL.PSX), -
/   (IPATH='<posix_install_path>')
```

In principle, the release items can be randomly renamed if this is taken into consideration during subsequent use. If the product files exist under a different userid and/or under a different name, the installation must take place for the product variant CPP in the following general form:

```
/CALL-PROCEDURE (<sinprc_name>,INSTALL.PSX), -
/   (IPATH='<posix_install_path>',SINLIB=<sinlib_name>, -
/    SYSMES=<sysmes_name>,SINPRC=<sinprc_name>)
```

The names of the release items can be normal file names with/without an userid and with/without a pubset specification.

The specified installation path may not exist, but the caller must be authorized to create it. If the path name is a relative path name, it is created in the HOME directory of the user.

3.3.1.6.3 Errors during the installation

If the installation is ended with an error, it is possible for the file SINLIB to already have been modified. It must be put back into the original status before another attempt at installation is made.

3.3.1.6.4 Preloadable subsystem

The ability to preload is not planned for privately installed POSIX compilers. What is worse is that there is potential for conflict here. It is currently not possible in POSIX (analog to SHARE-SCOPE=*NONE in SDF) to prevent the compiler from linking during the start with the preloaded subsystem of a different public installation with a suitable version, but possibly with a different correction version.

3.3.1.6.5 Use

The C/C++ commands are accessible via <posix_install_path>/bin. The call can be achieved by specifying the explicit command paths or by entering the respective path <posix_install_path>/bin in the command search path of the caller.

3.3.1.6.6 Deinstallation

Deinstallation of the POSIX part of a private installation is done by simply deleting the POSIX installation directory and the relevant release items in the BS2000, unless the latter are still being used for a private BS2000 installation.

3.3.1.7 Installation of the II-Update tool

The II-Update tool that is supplied with the SDF compiler is not IMON-compliant and must therefore almost always be adapted to the actual compiler installation using a special procedure. This applies for all installation types, and the only exception is a compiler installation under the system default userid (mostly TSOS) while retaining the original name of the release items.

Special provisions are also required for the call.

3.3.1.7.1 Prerequisites

The modification should take place under the same userid, under which the C/C++ installation is to take place that is being modified.

The release items of C/C++ must be accessible on this or another userid for the caller, the caller must above all also have write authorization for the files SYSSPR and SYSSDF.IU.USER. The contents of these files must also be in their original status and may not already have been changed by a prior modification.

The release items that are absolutely necessary for modification of the C/C++ installation are as follows (specified here with their original names):

```
SINPRC.CPP.040
SYSLNK.CPP.040
SYSSPR.CPP.040.IU
SYSSDF.CPP.040.IU.USER
```

3.3.1.7.2 Performing the modification

If the product files exist under their original names on the caller’s userid, the modification can then be performed with the command

```
/CALL-PROCEDURE (SINPRC.CPP.040,INSTALL.IU).
```

In principle, the release items can be randomly renamed if this is taken into consideration during subsequent use. If the product files exist under a different userid and/or under a different name, the modification must take place in the following general form:

```
/CALL-PROCEDURE (<sinprc_name>,INSTALL.IU), -
/   (SYSLNK=<syslnk_name>,SYSSDF=<sdfuser_iu_name>, -
/   SYSSPR=<sysspr_name>,SINPRC=<sinprc_name>)
```

The names of the release items can be normal file names with/without an userid and with/without a pubset specification.

3.3.1.7.3 Errors during modification

If the modification is ended with an error, it is possible for the file SYSSPR and/or SYSSDF.IU.USER to already have been modified. It must be put back into the original status before another attempt is made.

3.3.1.7.4 Use

After the modification has been completed, the II-Update tool can be used. It should be noted that the syntax file of the tool must be activated at least once per session. Then the start command can be called.

A tool call for an installation using the caller userid with the original names could be as follows:

```
/MOD-SDF-OPT *ADD(SYSSDF.CPP.040.IU.USER)
/START-II-UPDATE <command_parameter>
```

If the product files are available under a different userid and/or under a different name, the call of the tool can e.g. take place in the following form:

```
/MOD-SDF-OPT *ADD(<sdfuser_iu_name>)
/START-II-UPDATE <command_parameter>
```

3.3.2 Use of other products

Other products are required on the appropriate system for the normal procedure of the compiler. Regardless of the compiler installation type it determines the location of the product files of the required products per IMON and if unsuccessful, it then assumes a standard installation of the respective product under TSOS. If the compiler cannot find the required external product files in this way or the required minimum version is not available, the compiler cannot perform specific sub-functions.

3.4 Product use

3.4.1 C++ source and object compatibility

The C++ language variant Cfront V3.1.3, which is the only one possible up to V2.2, is set as of V4.0 with language mode *V2 (MODIFY-SOURCE-PROPERTIES LANGUAGE=CPLUSPLUS(MODE=*V2) or in POSIX with -X d). Objects compiled in this way can be mixed in an application with C++ objects that were generated with C/C++ < V3.

The additional C++ language variant von V3.x (ANSI or STRICT ANSI), is set as of V4.0 with language mode *V3 (MODIFY-SOURCE-PROPERTIES LANGUAGE=CPLUSPLUS(MODE=*V3) or (MODE=*V3, STRICT=*YES) or in POSIX with -X w or e). Objects compiled in this way can be mixed in an application with C++ objects that were generated with C/C++ < V4.

*2 The C++2020 mode is used as the default setting in V4.0A40 and later versions.
*V2 (Cfront), *V3 (ANSI) and C++2017 modes cannot be mixed:
*V2 C++- *V3 C++- and C++2017 objects cannot be linked together!

*2 The C++2017 mode and C++2020 mode is binary compatible.

3.5 Discontinued functions (and those to be discontinued)

The following functions of version 3.2 have been either removed in V4.0 or will be discontinued:

3.5.1 Obsolete functions

None.

3.5.2 Functions to be discontinued

3.5.2.1 POSIX options

The following POSIX options will no longer be guaranteed in later versions:

Up to previous versions	as of V4.0 replaced by
-F R<=...>	-F loopunroll<,...>
-OI<...>	-F i<...>
-R Tc	-K statistics
-R Ti	-K no_integer_overflow
-R Tp	-K no_prompting
-R Ts=...	-K stacksize=...
-W 0	-R min_weight,errors
-W 1	-R min_weight,warnings
-W 2	-R min_weight,notes
-X l	-N ls oder -N source_error
-X la	-N la oder -N object
-X li=0	-K include_all
-X li=1	-K include_name
-X li=2	-K include_user
-X lign	-K pragmas_ignored
-X llpp=...	-N output,,,...
-X lm	-N data_allocation_map
-X lp	-N project
-X lr	-N output,,for_rotation_print
-X ls	-N ls oder -N source_error
-X lx	-N lx oder -N cross_reference
-X C	-K subcall_lab
-X EC	-K calendar_etpnd
-X EJ	-K julian_etpnd
-X IFN	-N cif,project
-X IFX	-N cif,cross_reference
-X II	-K ilcs_opt
-X IO	-K ilcs_out
-X L	-K enum_long
-X LLMK	-K llm_keep
-X LLML	-K llm_case_lower
-X M	-K external_multiple
-X OD=...	-N output,...
-X RC	-K roconst
-X RS	-K rostr
-X S	-K share
-X U	-K external_unique
-X W	-K workspace_stack

3.6 Incompatibilities

3.6.1 Initialization

As of C/C++ V4.0 the initialization of arrays and structures with non-constant values is only possible in C2011 mode.

*4 ### 3.6.2 Abbreviation //MO-LI is now ambiguous

*4 The abbreviation //MO-LI or //MODIFY-LI is no longer unique. Additionally to
*4 //MODIFY-LISTING-PROPERTIES, there is now //MODIFY-LIBRARY-VERSION.

3.7 Restrictions

3.7.1 AID errors with constructors of length 0

Empty constructors in C++ sources may lead to AID errors in case of debugging:
AID0252 AID error in module 56 : RTC 0E (CMD: TRACE)

3.7.2 Peculiarities in Cfront C++ language mode

For each function in C++, an external name is generated that also contains the parameter types in encrypted form. Unfortunately, errors occurred in V2.2 that are now also incorporated in the present version for compatibility reasons. They only occur in *V2 (Cfront) C++ mode (//MODIFY-SOURCE-PROPERTIES LANGUAGE=CPLUS-PLUS(MODE=*V2) or. -X d).

The following three function pairs each receive the same external name and therefore lead to duplicates during binding:

```
f(char)
f(signed char)
```

C/C++ V2.2 did not know signed and both functions are therefore mapped to the same name. This only affects 'signed char'.

```
f(char (*x)[15])
f(char (*x)[18])
```

The same names since the array dimensions are not considered.

```
f(const c *)
f(c *)
```

if, for example, c was declared as 'typedef char c;': a const qualifier on a typedef remains invisible in the external name and both functions therefore receive the same name.

3.7.3 CFE1079 when using options IEEE or ASCII

Using the compile options FP-ARITMETICS = *IEEE (-K ieee_floats) or LITERAL-ENCODING = *ASCII / *ASCII-FULL (-K literal_ encoding_ascii / -K literal_encoding_ascii_full) may lead to an error 'CFE1079 ERROR ..: expected a type specifier' if the requirements described in the compiler user guide (C library functions must not be declared explicitly in the source but only by including the corresponding CRTE header) have not been met.

3.8 Procedure in the event of errors

Following error documentation is required for diagnostic purposes in the event of errors:

- brief description of error situation
- description how and if the error is reproducible
- options-, source-, error list with expanded user includes (LISTING-Option)
- runtime log (MSG=FH)
- pre-processor output/source
- object list
- binder list
- input/output files
- expected results
- dump file (if a dump occurred)
- brief description of run

4 Hardware requirements

C/C++ V4.0 may be used on all business servers fulfilling the software requirements.