

# **What's New in LEF 5.8 C/C++ Programming Interface**

**Product Version 5.8**  
**May 2017**

© 2017 Cadence Design Systems, Inc. All rights reserved.  
Printed in the United States of America.

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

**Trademarks:** Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

**Restricted Print Permission:** This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

**Disclaimer:** Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

**Restricted Rights:** Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

---

# Contents

---

<u>About This Manual</u> .....	3
<u>How This Document Is Organized</u> .....	3
<u>Related Documents</u> .....	3

## 1

<u>New Features</u> .....	5
<u>General LEF Reader Changes</u> .....	6
<u>LEF Reader Setup and Control Routines</u> .....	7
<u>LEF Reader Callback Routines</u> .....	7
<u>LEF Reader Classes</u> .....	7

## 2

<u>Changed Features</u> .....	9
<u>Layer Routines</u> .....	10
<u>lefiGeometries</u> .....	10
<u>lefiGeomRect</u> .....	11
<u>lefiGeomRectIter</u> .....	11
<u>lefiGeomPath</u> .....	11
<u>lefiGeomPathIter</u> .....	11
<u>lefiGeomPolygon</u> .....	12
<u>lefiGeomPolygonIter</u> .....	12
<u>lefiGeomVia</u> .....	12
<u>lefiGeomViaIter</u> .....	12
<u>lefiLayer</u> .....	13
<u>lefiViaLayer</u> .....	13
<u>lefiVia</u> .....	14

## What's New in LEF 5.8 C/C++ Programming Interface

---

---

# About This Manual

---

This document provides information on new and changed features for version 5.8 of the C and C++ application programming interface (API) used to read and write Cadence® Library Exchange Format (LEF) files.

## How This Document Is Organized

This *What's New* document is organized into the following chapters:

- **New Features**

This chapter describes features that were added since version 5.7 of the LEF API. New features are those that introduce new functionality into the LEF API. Any enhancements made to existing statements to support a new feature are also described in this chapter.

- **Changed Features**

This chapter discusses features that were changed since version 5.7 of the LEF API. Changed features include such things as changes in default behavior, changes in whether keywords and statements are required, and any other changes that do not reflect new functionality.

## Related Documents

The following documents provide detailed information about LEF and DEF, and the LEF and DEF application programming interfaces.

- [DEF C/C++ Programming Interface \(Open Licensing Program\)](#)
- [LEF C/C++ Programming Interface \(Open Licensing Program\)](#)
- [LEF/DEF Language Reference](#)
- [What's New in DEF C/C++ Programming Interface](#)
- [What's New in LEF/DEF](#)

7/10/17

# **What's New in LEF 5.8 C/C++ Programming Interface**

## About This Manual

---

---

## New Features

---

This chapter describes the new features that were added in this release of the LEF application programming interface.

- [General LEF Reader Changes](#) on page 6
- [LEF Reader Setup and Control Routines](#) on page 7
- [LEF Reader Callback Routines](#) on page 7
- [LEF Reader Classes](#) on page 7

## General LEF Reader Changes

The following changes were made in the latest version of the parser:

- **Elimination of static data:** In the latest version, the parser architecture has changed from C style-based to C++ style-based. In the previous version of the parser, most of the parser data were stored in static variables and the data lifecycle was based on initializers and cleaners. The new architecture places data in data singletons and uses C++ constructors and destructors. The switch to the C++ architecture has improved the parser re-enterability, made the data flow more robust, and helped clean multiple memory leaks in the parser code.
- **Introduction of parsing sessions:** In the previous version, data were stored in static variables and, therefore, were retained across all parsing cycles. This meant that if a property was defined once, it continued to be defined in the next LEF file reads. In some applications, this feature was actively used. In others, it disturbed expected application behavior. To address this issue, the latest version of the parser introduces two modes of files processing - compatibility mode and session-based mode.
  - Compatibility mode (session-less mode) - This mode is compatible with the old parser behavior. You can call the parser initialization once with `lefrInit()`, adjust parsing settings and initialize the parser callbacks any time. The properties in `PROPERTYDEFINITIONS` sections will be active in all subsequent file reads.
  - Session-based mode - This mode introduces the concept of a parsing session – the parser configuration settings will be active during the session time and will be cleaned on its end. The parsing session also controls `PROPERTYDEFINITIONS` data. Property definitions remain active throughout the parsing session time and are cleaned at the end of the session. The session-based mode does not require calling callbacks and configuration unset functions - all callbacks and properties will be set to defaults by `lefrClear()` or the next session `lefrInitSession()` call.

By default, the LEF parser works in the compatibility mode. To activate the session-based mode, you must use `lefrInitSession()` instead of `lefrInit()`.

**Note:** Currently, the compatibility mode is used for all old applications for which code has not been adjusted. The `lef2oa` and `def2oa` translators have adjusted to use the session-based parsing mode.

For more information, see [“LEF Reader Working Modes”](#) in the *LEF 5.8 C/C++ Programming Interface (Open Licensing Program)*.

- **Long DEF Files Support:** In this version, the LEF line counter switched to 64-bit integer type, making it possible to process files with more than two billion lines.



## LEF Reader Setup and Control Routines

The following reader setup and class routines were added in this release:

- [lefrInitSession](#)
- [lefrClear](#)
- [lefrRegisterLef58Type](#)

For more information, see “[LEF Reader Setup and Control Routines](#)” in the *LEF 5.8 C/C++ Programming Interface (Open Licensing Program)*.

## LEF Reader Callback Routines

The following reader callback routines were added in this release:

- [lefrFixedMaskCbK](#)
- [lefrMacroFixedMaskCbK](#)
- [lefrSetMacroForeignCbK](#)
- [lefrSetMacroSiteCbK](#)
- [lefrUnsetMacroForeignCbK](#)
- [lefrUnsetMacroSiteCbK](#)

For more information on reader callback routines, see “[LEF Reader Callback Routines](#)” in the *LEF 5.8 C/C++ Programming Interface (Open Licensing Program)*.

## LEF Reader Classes

The following LEF Reader classes were added in this release:

- [lefiMacroSite](#)
- [lefiMacroForeign](#)

For more information on reader classes, see “[LEF Reader Classes](#)” in the *LEF 5.8 C/C++ Programming Interface (Open Licensing Program)*.

## **What's New in LEF 5.8 C/C++ Programming Interface**

### **New Features**

---

---

## Changed Features

---

This chapter describes the features that were changed in this release of the LEF application programming interface.

- Layer Routines on page 10

## Layer Routines

The following syntax has been added to the listed layer routines.

### lefiGeometries

```
int colorMask;  
void addPath(int colorMask);  
void addPathIter(int colorMask);  
void addRect(int colorMask);  
void addRectIter(int colorMask);  
void addPolygon(int colorMask);  
void addPolygonIter(int colorMask);  
void addVia(int viaMasks);  
void addViaIter(int viaMasks);
```

These are described below:

`colorMask`                      Defines the color mask number for the `GeomRect` structure.

`addPathIter(int colorMask)`  
                                 Adds the color mask number to the `lefiGeomPath` structure.  
                                 The default value is 0.

`addPathIter(int colorMask)`  
                                 Adds the color mask number to the `lefiGeomPathIter`  
                                 structure. The default value is 0.

`addRectIter(int colorMask)`  
                                 Adds the color mask number to the `lefiGeomRectIter`  
                                 structure. The default value is 0.

`addPolygon(int colorMask)`  
                                 Adds the color mask number to the `lefiGeomPolygon`  
                                 structure. The default value is 0.

`addPolygonIter(int colorMask)`  
                                 Adds the color mask number to the `lefiGeomRectIter`  
                                 structure. The default value is 0.

`addVia(int viaMasks)`

## What's New in LEF 5.8 C/C++ Programming Interface

### Changed Features

---

Adds the via mask number to the `lefiGeomVia` structure. The default value is 0.

```
addViaIter(int viaMasks)
```

Adds the via mask number to the `lefiGeomViaIter` structure. The default value is 0.

For more information, see [“lefiGeometries”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### **lefiGeomRect**

```
int colorMask;
```

Defines the color mask number for the `GeomRect` struct. The default value is 0.

For more information, see [“lefiGeomRect”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### **lefiGeomRectIter**

```
int colorMask;
```

Defines the color mask number for the `GeomRectIter` struct. The default value is 0.

For more information, see [“lefiGeomRect”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### **lefiGeomPath**

```
int colorMask;
```

Defines the color mask number for the `GeomPath` struct. The default value is 0.

For more information, see [“lefiGeomPath”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### **lefiGeomPathIter**

```
int colorMask;
```

Defines the color mask number for the `GeomPathIter` struct. The default value is 0.

## What's New in LEF 5.8 C/C++ Programming Interface

### Changed Features

---

For more information, see [“lefiGeomPathIter”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### lefiGeomPolygon

```
int colorMask;
```

Defines the color mask number for the `GeomPolygon` struct. The default value is 0.

For more information, see [“lefiGeomPolygon”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### lefiGeomPolygonIter

```
int colorMask;
```

Defines the color mask number for the `GeomPolygonIter` struct. The default value is 0.

For more information, see [“lefiGeomPolygonIter”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### lefiGeomVia

```
int topMaskNum;  
int cutMaskNum;  
int bottomMaskNum;
```

Indicates the top, bottom, and cut mask numbers for the `GeomVia` struct. The default value is 0.

For more information, see [“lefiGeomVia”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### lefiGeomVialter

```
int topMaskNum;  
int cutMaskNum;  
int bottomMaskNum;
```

Indicates the top, bottom, and cut mask numbers for the `GeomViaIter` struct. The default value is 0.

For more information, see [“lefiGeomVialter”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

## What's New in LEF 5.8 C/C++ Programming Interface

### Changed Features

---

#### lefiLayer

```
void setMask(int num);  
int hasMask() const;  
Int mask() const;
```

These are described below:

<code>setMask(int num)</code>	Sets the color mask number on the layer.
<code>hasMask()</code>	Checks whether the layer has a color mask assigned to it or not.
<code>mask()</code>	Returns the color mask number of the layer.

For more information, see [“lefiLayer”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

#### lefiViaLayer

```
void addRect(int mask,  
             double xl,  
             double yl,  
             double xh,  
             double yn);  
void addPoly(int mask,  
             lefiGeometries* geom);  
int rectColorMask(int index);  
int polyColorMask(int index);
```

These are described below:

<code>addRect(int mask)</code>	Adds the color mask number to the rectangle inside the via. The default value is 0.
<code>addPoly(int mask)</code>	Adds the color mask number to the polygon. The default value is 0.
<code>rectColorMask(int index)</code>	Returns the color mask number on the rectangle.
<code>polyColorMask(int index)</code>	Returns the color mask number on the polygon.

## What's New in LEF 5.8 C/C++ Programming Interface

### Changed Features

---

For more information, see [“lefiViaLayer”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.

### lefiVia

```
void addRectToLayer(int mask,
    double xl,
    double yl,
    double xh,
    double yh);
void addPolyToLayer(int mask,
    lefiGeometries* geom);
int rectColorMask(int layerNum,
    int rectNum);
int polyColorMask(int layerNum,
    int rectNum);
```

These are described below:

`addRectToLayer(int mask)`

Adds the color mask of the via rectangle to the layer. The default value is 0.

`addPolyToLayer(int mask)`

Adds the color mask of the via polygon to the layer. The default value is 0.

`rectColorMask(int layerNum, int rectNum)`

Returns the color mask number of the indexed rectangle inside the via for that layer.

`polyColorMask(int layerNum, int rectNum)`

Returns the color mask number of the indexed polygon inside the via for that layer.

For more information, see [“lefiVia”](#) in the *LEF C/C++ Programming Interface (Open Licensing Program)*.