インテル® oneAPI ツールキット 2023 における DPC++ ランタイム環境変数

この記事は、GitHub* の LLVM で公開されている 2023 年 4 月 12 日現在の「Environment Variables」を、インテル社の許可を得て iSUS (IA Software User Society) が翻訳した日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。

このドキュメントでは、DPC++ コンパイラーとランタイムに影響する環境変数について説明します。

DPC++ ランタイムの制御

環境変数	値	説明
ONEAPI_DEVICE_SELECTOR	こちらを参照	このデバイス選択環境変数によって、SYCL* ベースのアプリケーションの実行時に使用するデバイスの選択を制御できます。デバイスを特定のタイプ (GPU やアクセラレーター) またはバックエンド (レベルゼロや OpenCL*) に制限するのに役立ちます。このデバイス選択のメカニズムは、SYCL_DEVICE_FILTER を置き換えるものです。ONEAPI_DEVICE_SELECTOR の構文は OpenMP* と共有されており、サブデバイスを選択することもできます。完全な説明は、こちらを参照してください。
SYCL_DEVICE_FILTER (deprecated)	backend:device_type:device_num	代わりに ONEAPI_DEVICE_SELECTOR 環境変数を使用してください。SYCL_DEVICE_FILTER の詳しい説明は、以下の「SYCL_DEVICE_FILTER」の節を参照してください。
SYCL_DEVICE_ALLOWLIST	こちらを参照	指定するパターンに一致しないデバイスを除外します。BackendName には、host、opencl、level_zero または cuda を指定できます。DeviceType には、host、cpu、gpu または acc を指定できます。DeviceVendorId には 16 進形式の uint32_t を指定します (0xXYZW)。DriverVersion、PlatformVersion、DeviceName および PlatformName には、正規表現を指定できます。括弧などの特殊文字はエスケープする必要があります。DPC++ ランタイムは、上記で指定された値と正規表現を満たすデバイスのみを選択します。複数のデバイスを指定するには、「|」(パイプ記号: 縦棒) を使用します。
SYCL_DISABLE_PARALLEL_FOR_RANGE_ROUNDING	任意 (*)	parallel_for 呼び出し範囲の自動切り上げを無効にします。

環境変数	値	説明
SYCL_CACHE_DIR	パス	永続キャッシュ・ルート・ディレクトリーへのパスを指定します。デフォルト値は、Windows* では %AppData%\libsycl_cache です。Linux* では $XDG_CACHE_HOME/libsycl_cache ですが、XDG_CACHE_HOME が設定されていない場合は $HOME/.cache/libsycl_cache になります。環境変数が設定されていない場合、SYCL* 永続キャッシュは無効になります。
SYCL_CACHE_DISABLE_PERSISTENT (deprecated)	任意 (*)	効果はありません。
SYCL_CACHE_PERSISTENT	整数	永続的デバイスのコンパイル済みコードのキャッシュを制御します。1 に設定されると ON になり、0 では OFF になります。キャッシュが有効になると、SYCL* ランタイムは JIT コンパイル済バイナリーをキャッシュして再利用します。デフォルトは OFF です。
SYCL_CACHE_EVICTION_DISABLE	任意 (*)	変数が設定されていると、キャッシュの排出を OFF に切り替えます。
SYCL_CACHE_MAX_SIZE	正の整数	キャッシュされたイメージの合計サイズがメガバイト単位の値 (デフォルト: 8GB の場合は 8192) を超えると、キャッシュの排出がトリガーされます。サイズベースのキャッシュ排出を無効にするには、0 に設定します。
SYCL_CACHE_THRESHOLD	正の整数	キャッシュ排出のしきい値 (日数) を設定します。デフォルト値は 7 (1 週間) です。日数ベースのキャッシュ排出を無効にするには、0 に設定します。
SYCL_CACHE_MIN_DEVICE_IMAGE_SIZE	正の整数	ディスクアクセスは JIT コンパイルよりも時間を要する場合があるため、ディスクにキャッシュするデバイス・コード・イメージの最小サイズをバイト単位で指定します。デフォルト値は 0 であり、すべてのイメージをキャッシュします。
SYCL_CACHE_MAX_DEVICE_IMAGE_SIZE	正の整数	キャッシュするデバイスイメージの最大サイズをバイト単位で指定します。カーネルが大きすぎるとディスクに負荷がかかりすぎることがあります。デフォルト値は 1GB です。
SYCL_ENABLE_DEFAULT_CONTEXTS	1 または 0	SYCL* ランタイムでデフォルト・プラットフォーム・コンテキストの生成を有効 (「1」) または無効 (「0」) に設定します。それぞれのプラットフォームのデフォルト・コンテキストには、プラットフォーム内のすべてのデバイスが含まれます。詳細は、プラットフォームのデフォルト・コンテキスト (英語) を参照してください。Linux* ではデフォルトで有効ですが、Windows* では無効です。
SYCL_RT_WARNING_LEVEL	正の整数	警告レベルが高いほど、ランタイム・ライブラリーが出力する警告とパフォーマンスのヒントが増加します。デフォルトは「0」であり、ランタイム・ライブラリーから警告やパフォーマンスのヒントは出力されません。1 に設定すると、デバイスランタイム/codegen からのパフォーマンスの警告が有効になります。1 より大きな値は、将来に向けて予約済みです。

環境変数	値	説明
SYCL_USM_HOSTPTR_IMPORT	整数	ゼロ以外の値を指定して機能を有効にします。ホストポインターで作成されたバッファーは、ホストデータが USM に昇格するため、データ転送のパフォーマンスが向上します。この機能を使用するには、SYCL_HOST_UNIFIED_MEMORY=1 に設定します。
SYCL_EAGER_INIT	整数	ゼロ以外の値を指定して機能を有効にします。その都度遅延初期化を行うのではなく、オブジェクトの構築時に初期化を行うよう SYCL* ランタイムに指示します。これは、ウォームアップ時に冗長な処理が行われる可能性がありますが、ホットでレポート可能なパスでは最速の実行を保証します。PI プラグインにも同様な振る舞いをするよう指示します。デフォルトは「0」です。
SYCL_REDUCTION_PREFERRED_WORKGROUP_SIZE	こちらを参照	リダクションの推奨 work-group サイズを制御します。
SYCL_ENABLE_FUSION_CACHING	1 または 0	カーネル融合の JIT コンパイルのキャッシュを有効 (「1」) または無効 (「0」) にします。同一カーネルが複数回融合される場合、キャッシュにより JIT コンパイル・パイプラインの繰り返し実行が回避されます。デフォルト値は 1 です。

(*) 注: 任意は、この環境変数が NULL 以外の値に設定されている場合に有効であることを意味します。

ONEAPI_DEVICE_SELECTOR

この環境変数を設定しなければ、現在のマシンにあるすべてのプラットフォームとデバイスが利用できます。デフォルトの選択は、それらデバイスのいずれかですが、通常、利用可能なレベルゼロ GPU デバイスが優先されます。ONEAPI_DEVICE_SELECTOR を使用してデバイスの選択を制限し、GPU サブデバイスまたはサブのサブデバイスを個々のデバイスとして識別できます。

この環境変数の構文は、次の BNF 文法に従います。

```
ONEAPI_DEVICE_SELECTOR = <selector-string>
<selector-string> ::= { <accept-filters> | <discard-filters> | <accept-filters>;<discard-filters> }
<accept-filters> ::= <accept-filter>[;<accept-filter>...]
<discard-filters> ::= <discard-filter>[;<discard-filter>...]
<accept-filter> ::= <term>
<discard-filter> ::= !<term>
<term> ::= <backend>:<devices>
<backend> ::= { * | level_zero | opencl | cuda | hip | esimd_emulator }  // case insensitive
<devices> ::= <device>[,<device>...]
<device> ::= { * | cpu | gpu | fpga | <num> | <num>.<num> | <num>.* | *.* | <num>.<num>.<num> | <num>.<num>.* | <num>.*.* | *.*.*  }  // 大文字小文字を区別しません
```

文法の各項目は、特定のバックエンドからデバイスのコレクションを選択します。デバイス名 cpu、gpu、および fpga は、対応するタイプのバックエンドからすべてのデバイスを選択します。バックエンドのデバイスは、数値インデックス (ゼロベース)、または * (バックエンドのすべてのデバイスを選択) を使用して選択できます。

ドット構文 (<num>.<num> など) を使用すると、1 つ以上の GPU サブデバイスが SYCL* ルートデバイスとしてアプリケーションに公開されます。例えば、1.0 では 2 番目のデバイスの最初のサブデバイスを SYCL* ルートデバイスとして公開します。<num>.* 構文は、特定のデバイスのすべてのサブデバイスを SYCL* ルートデバイスとして公開します。*.* 構文は、すべての GPU デバイスのすべてのサブデバイスを SYCL* ルートデバイスとして公開します。

通常、1 つ以上のアスタリスク (*) を持つ項目は、指定されたパターンのすべてのバックエンド、デバイス、またはサブデバイスと一致します。ただし、項目がいずれにも一致しない場合、警告が表示されます。例えば *:gpu は、すべてのバックエンドのすべての GPU デバイスに一致しますが、GPU デバイスのないバックエンドは無視され、どのバックエンドにも GPU デバイスがない場合は警告されます。同様に level_zero:*.* は、レベルゼロ・バックエンドのパーティション化が可能な GPU のすべてのサブデバイスに一致しますが、サブデバイスにパーティション化可能なレベルゼロ GPU デバイスがない場合は警告が表示されます。

デバイスのインデックスは 0 から始まり、バックエンド内でのみ一意です。そのため、level_zero:0 と cuda:0 は異なるデバイスとなります。利用可能なすべてのデバイスのインデックスを知るには、sycl-ls ツールを実行します。異なるバックエンドが同じハードウェアを異なる「デバイス」として公開することがあることに注意してください。例えば、level_zero と opencl バックエンドはどちらもインテル® GPU デバイスを公開します。

さらに、(数値インデックスやワイルドカードで) サブデバイスが選択された場合、パーティション化された追加レイヤーを指定できます。つまり、サブデバイスを選択することができます。サブデバイスと同様に、これにはピリオド (.)、ワイルドカード (*)、または数値インデックスであるサブデバイス指定子を使用できます。例えば、ONEAPI_DEVICE_SELECTOR=level_zero:0.*.* は、デバイス 0 をサブデバイスにパーティション化し、それぞれをサブのサブデバイスにパーティション化します。孫であるサブのサブデバイスのレンジは、アプリケーションで使用できる最終的なデバイスであり、デバイス 0 やその子パーティションはリストに含まれません。

最後に、文法内のフィルターは、用語で選択されるすべてのデバイスで実行されるアクションに関連すると考えることができます。アクションには、受け入れアクションと破棄アクションがあります。アクションに基づいて、フィルターは受け入れフィルターと破棄フィルターになります。<term> は受け入れフィルターを表わし、!<term> は破棄フィルターを表わします。基本的な用語は同じですが、一致するデバイスリストに対して異なるアクションを実行します。例えば、!opencl:* は、利用可能なデバイスリストから opencl バックエンドのすべてのデバイスを破棄します。破棄フィルターがある場合。それらはすべてのセレクター文字列の最後である必要があります。1 つまたは複数のフィルターがデバイスを受け入れ、そして 1 つまたは複数のフィルターがそのデバイスを破棄する場合、後者が優先されデバイスは利用できなくなります。これにより、CUDA* バックエンドを持つデバイスを除くすべての gpu デバイスを入れ代えれる、*:gpu;!cuda:* のようなセレクター文字列を利用できます。さらに、この環境変数の値に破棄フィルターのみが設定されると、サブデバイスおよびサブのサブデバイスを除くすべてのデバイスに一致する受け入れフィルターを暗黙的に含めることで、利用可能にしないデバイスリストのみを指定できるようになります。したがって、!*:cpu は cpu タイプのデバイスを除く全てのデバイスを受け入れ、opencl:*;!*:cpu は opencl バックエンドと cpu タイプのデバイスを除く、opencl バックエンドのすべてのデバイスを受け入れます。選択文字列の中で、以前のフィルターですでに省略されたデバイスを指定する場合でも、拒否フィルターを使用することは違法ではありません。これを行っても効果はありませんが、拒否されたデバイスは引き続き省略されます。

以下に、この環境変数の使用例を示します。

例	結果
`ONEAPI_DEVICE_SELECTOR=opencl:*`	OpenCL* デバイスのみ利用できます。
`ONEAPI_DEVICE_SELECTOR=level_zero:gpu`	ゼロレベル・プラットフォームの GPU デバイスのみ利用できます。
`ONEAPI_DEVICE_SELECTOR="opencl:gpu;level_zero:gpu"`	ゼロレベルと OpenCL* の両方の GPU デバイスを利用できます。セミコロンで区切られたエントリーを指定する場合、エスケープ (引用符など) が必要になることがあります。
`ONEAPI_DEVICE_SELECTOR=opencl:gpu,cpu`	OpenCL* プラットフォームの CPU および GPU デバイスのみ利用できます。
`ONEAPI_DEVICE_SELECTOR=opencl:0`	OpenCL* バックエンドのインデックス 0 のデバイスのみ利用できます。
`ONEAPI_DEVICE_SELECTOR=hip:0,2`	HIP* バックエンドのインデックスが 0 と 2 のデバイスのみ利用できます。
`ONEAPI_DEVICE_SELECTOR=opencl:0.*`	インデックス 0 を持つ OpenCL* デバイスのすべてのサブデバイスは、SYCL* ルートデバイスとして公開されます。そのほかのデバイスは利用できません。
`ONEAPI_DEVICE_SELECTOR=opencl:0.2`	インデックス 0 の OpenCL* デバイスの 3 番目のサブデバイス (ゼロベースのカウント 2) が、利用可能な唯一のデバイスになります。
`ONEAPI_DEVICE_SELECTOR=level_zero:*,*.*`	2 つの異なる方法でゼロレベルデバイスをアプリケーションに公開します。各デバイス (別名: カード) は、SYCL* ルートデバイスとして公開され、それぞれのサブデバイスも SYCL* ルートデバイスとして公開されます。
`ONEAPI_DEVICE_SELECTOR="opencl:*;!opencl:0"`	インテックス 0 のデバイスを除くすべての OpenCL* デバイスを利用できます。
`ONEAPI_DEVICE_SELECTOR="!*:cpu"`	CPU デバイスを除くすべてのデバイスを利用できます。

注:

- バックエンドの引数は省略できません。引数がないとエラーがスローされます。
- さらに、バックエンドの後にはコロン (:) と少なくとも 1 つのデバイス指定子が必要です。デバイス指定子がないとエラーがスローされます。
- サブデバイスとサブのサブデバイスでは、親デバイスはパーティション化をサポートする必要があります (`info::partition_property::partition_by_affinity_domain` および `info::partition_affinity_domain::next_partitionable`。精度の定義については、SYCL* 2020 仕様を参照してください)。インテル® GPU の場合、サブデバイスおよびサブのサブデバイス構文を使用して、タイルまたは CCS をルートデバイスとして SYCL* アプリケーションに公開できます。サブデバイス、サブのサブデバイス、タイル、および CSS 間の実際のマッピングは、ハードウェア固有です。
- セミコロン (;) と感嘆符 (!) は、多くのシェルで特殊文字として扱われるため、選択文字列にこれらを含める場合、文字列を引用符で囲む必要があります。

SYCL_DEVICE_ALLOWLIST

デバイスとドライバーバージョンをリストします:
`BackendName:XXX,DeviceType:YYY,DeviceVendorId:0xXYZW,DriverVersion:{{X.Y.Z.W}}`。
`PlatformVersion`、`DeviceName`、および `PlatformName` が含まれることもあります。表示されるプロパティーの順序は固定ではありません。

SYCL_DEVICE_FILTER

この環境変数は、SYCL* ランタイムがシステムデバイスのサブセットのみを使用するように限定します。この環境変数を設定すると、すべてのデバイスクエリー関数 (`platform::get_devices()` および `platform::get_platforms()`) とすべてのデバイスセレクターに影響します。

この環境変数の値は、カンマで区切ったフィルターのリストです。それぞれのフィルターは、「backend:device_type:device_num」(引用符なし) 形式のトリプルです。トリプルの各要素はオプションですが、各フィルターには少なくとも 1 つの値が必要です。`backend` には以下を指定できます。

- `host` (非推奨)
- `level_zero`
- `opencl`
- `cuda`
- `hip`
- `esimd_emulator`
- `*`

`device_type` には以下を指定できます。

- `host` (非推奨)
- `cpu`
- `gpu`
- `acc`
- `*`

`device_num` は、sycl-ls ユーティリティー・ツールからのデバイス列挙をインデックス指定する整数であり、列挙の最初のデバイスはそれぞれのバックエンドでインテックス 0 となります。例えば、`SYCL_DEVICE_FILTER=2` は、すべてのバックエンドからインデックス 2 を持つすべてのデバイスを返します。複数のデバイスがこのデバイス番号を満たす場合 (例: GPU および CPU デバイスにデバイス番号 2 を割り当て可能)、`default_selector` は評価値が最も高いデバイスを選択します。`SYCL_DEVICE_ALLOWLIST` が設定されている場合、デバイスを列挙する前に適用され、`device_num` 値に影響します。

フィルターにトリプルの 3 つの要素が含まれると仮定すると、指定されたバックエンドから取得され、指定されたデバイスタイプを持ち、かつ指定されたデバイス・インデックスを持つデバイスのみが選択されます。複数のフィルターが指定されると、ランタイムはすべてのフィルターによって選択されたデバイスの結合に制限されます。

フィルター処理されたデバイスリストにセレクターを満たすデバイスが含まれていない場合、すべてのデバイスセレクターが例外をスローすることに注意してください。例えば、`SYCL_DEVICE_FILTER=cpu` にすると `gpu_selector()` は例外をスローします。`SYCL_DEVICE_FILTER` は、指定されたプラグインのみを SYCL* ランタイムにロードすることも制限します。特に、`SYCL_DEVICE_FILTER=level_zero` に設定すると、SYCL* ランタ

イムはその時点で CPU デバイスをサポートしない `level_zero` バックエンドのみをロードするため、`cpu_selector` が例外をスローします。複数のデバイスがフィルターの条件を満たす場合 (例: `SYCL_DEVICE_FILTER=gpu`)、そのうち 1 つだけが選択されます。

SYCL_REDUCTION_PREFERRED_WORKGROUP_SIZE

この環境変数は、指定されたデバイスタイプでリダクションのため推奨される work-group サイズを制限します。この変数を設定すると、環境変数の値に含まれるタイプのデバイスで、明示的な work-group サイズを持たないすべてのリダクションに影響します。

この環境変数の値は、カンマで区切った 1 つ以上のリストです。それぞれは、「`device_type:size`」(引用符なし) 形式のペアです。`device_type` には以下を指定できます。

- `cpu`
- `gpu`
- `acc`
- `*`

`size` は 0 より大きな正の整数です。

`device_type:size` の場合、`device_type` 要素は構成が適用されるデバイスタイプを指定します。つまり、`cpu` は CPU デバイス用、`gpu` は GPU デバイス用、そして `acc` はアクセラレーター・デバイス用です。`device_type` が `*` である場合、構成は該当するすべてのデバイスタイプに適用されます。`size` は、`device_type` で指定されるタイプのデバイスに使用される推奨 work-group サイズを示します。

リダクションがキューに投入されるデバイスの `info::device::max_work_group_size` が、この環境変数で設定される値よりも小さい場合、そのデバイスの `info::device::max_work_group_size` 値が代わりに使用されます。

以下の場合、リダクション・カーネルの送信中に `sycl::errc::invalid` コードの `sycl::exception` がスローされます。

- いずれの構成でも指定されたデバイスタイプが有効な値を持たない場合。
- いずれかの構成で指定された推奨 work-group サイズが有効な整数値でない場合。
- いずれかの構成で指定された推奨 work-group サイズが 0 より大きい整数値でない場合。
- 構成にデリミター (:) がない場合。

この環境変数が設定されていない場合、リダクションに推奨される work-group サイズは実装依存です。

同一リスト内で競合する構成タプルでは、最後のエントリーが優先されることに注意してください。例えば、`cpu:32,gpu:32,cpu:16` のリストは、リダクションの推奨 work-group サイズを GPU では 32、CPU では 16 に設定します。これは、`*` にも当てはまります。例えば、`cpu:32,*:16` はすべてのデバイスでリダクションの推奨 work-group サイズを 16 に設定しますが、`*:16,cpu:32` の場合は推奨 work-group サイズを CPU では 32 に、それ以外は 16 に設定します。

DPC++ レベルゼロプラグインの制御

環境変数	値	説明
SYCL_ENABLE_PCI	整数	1 に設定すると、レベルゼロ・バックエンドの利用時に GPU PCI アドレスが取得できるようになります。デフォルトは 0 です。
SYCL_PI_LEVEL_ZERO_DISABLE_USM_ALLOCATOR	任意 (*)	レベルゼロプラグインで USM 割り当てを無効にします (メモリー要求はレベルゼロランタイムに直接送られます)。
SYCL_PI_LEVEL_ZERO_TRACK_INDIRECT_ACCESS_MEMORY	任意 (*)	レベルゼロプラグインで、カーネルの間接アクセスと対応するメモリー割り当ての遅延リリースを有効にします。

(*) 注: 任意は、この環境変数が NULL 以外の値に設定されている場合に有効であることを意味します。

DPC++ CUDA* プラグインの制御

環境変数	値	説明
SYCL_PI_CUDA_MAX_LOCAL_MEM_SIZE	整数	ローカルメモリーの割り当て最大サイズをバイト単位で指定します。値がデバイスの能力を上回ると、sycl::runtime_error がスローされます。完全なエラーメッセージを取得するには、SYCL_RT_WARNING_LEVEL=2 に設定します。SYCL_PI_CUDA_MAX_LOCAL_MEM_SIZE のデフォルト値は、ハードウェアごとに異なります。

DPC++ HIP プラグインの制御

環境変数	値	説明
SYCL_PI_HIP_MAX_LOCAL_MEM_SIZE	整数	ローカルメモリーの割り当て最大サイズをバイト単位で指定します。値がデバイスの能力を上回ると、sycl::runtime_error がスローされます。完全なエラーメッセージを取得するには、SYCL_RT_WARNING_LEVEL=2 に設定します。SYCL_PI_HIP_MAX_LOCAL_MEM_SIZE のデフォルト値は、ハードウェアごとに異なります。

ツール変数

環境変数	値	説明
INTEL_ENABLE_OFFLOAD_ANNOTATIONS	任意 (*)	SYCL* ランタイムの ITT アノテーションのサポートを有効にします。この変数は、ITT アノテーションをサポートするツールでのみ使用する必要があります。
XPTI_FRAMEWORK_DISPATCHER(**)	ディスパッチャー・ライブラリーへのパス	XPTI インストルメント・ディスパッチャー・フレームワーク・ライブラリーをロードします。詳細は、XPTI フレームワークのドキュメント (英語) を参照してください。
XPTI_TRACE_ENABLE(**)	1、true、0、false	XPTI のインストルメンテーションを有効にします。詳細は、XPTI フレームワークのドキュメント (英語) を参照してください。
XPTI_SUBSCRIBERS(**)	カンマ区切りのサブスクライバー・ライブラリーのリスト	XPTI サブスクライバーをロードします。詳細は、XPTI フレームワークのドキュメント (英語) を参照してください。

(*) 注: 任意は、この環境変数が NULL 以外の値に設定されている場合に有効であることを意味します。
(**) 注: これらの変数は XPTI フレームワークから取得されます。

DPC++ ランタイム向けのデバッグ変数

⚠ **警告:** 以下の環境変数は、DPC++ コンパイラーとランタイムの開発とデバッグに使用されます。これらのセマンティクスは変更される可能性があります。製品化するコードでは、これらの変数に依存しないでください。

環境変数	値	説明
SYCL_PI_TRACE	こちらを参照	PI で指定されたレベルのトレースを有効にします。
SYCL_QUEUE_THREAD_POOL_SIZE	正の整数	キューのスレッドプール内のスレッド数を指定します。
SYCL_DEVICELIB_NO_FALLBACK	任意 (*)	デバイス・ライブラリー・イメージの読み込みとリンクを無効にします。
SYCL_PRINT_EXECUTION_GRAPH	こちらを参照	実行グラフを DOT テキストファイルに出力します。
SYCL_DISABLE_EXECUTION_GRAPH_CLEANUP	任意 (*)	キューに投入された (またはホストタスクの場合は終了した) 非リーフ・コマンド・ノードの定期的なクリーンアップを無効にします。無効にすると、コマンドノードは、使用された最後に残っているメモリー・オブジェクトの破棄中にのみクリーンアップされます。
SYCL_DISABLE_POST_ENQUEUE_CLEANUP (非推奨)	任意 (*)	代わりに SYCL_DISABLE_EXECUTION_GRAPH_CLEANUP を使用してください。
SYCL_DEVICELIB_INHIBIT_NATIVE	スペースで区切られたデバイス・ライブラリー拡張子の文字列	このオプションにリストされている devicelib 拡張のデバイス・ネイティブ・サポートに依存してはなりません。
SYCL_PROGRAM_COMPILE_OPTIONS	有効な OpenCL* コンパイルオプションの文字列	すべてのプログラムのコンパイルオプションをオーバーライドします。

環境変数	値	説明
SYCL_PROGRAM_LINK_OPTIONS	有効な OpenCL* リンクオプションの文字列	すべてのプログラムのリンクオプションをオーバーライドします。
SYCL_USE_KERNEL_SPV	SPIR-V* バイナリーへのパス	指定したファイルからデバイスイメージをロードします。ランタイムがファイルをリードできない場合、sycl::runtime_error 例外がスローされます。
SYCL_DUMP_IMAGES	任意 (*)	デバイスのイメージバイナリーをファイルにダンプします。SYCL_USE_KERNEL_SPV が設定されている場合、効果はありません。
SYCL_HOST_UNIFIED_MEMORY	整数	実行グラフビルダーに対し、ホスト統合メモリーのサポートまたは非サポートを強制します。0 に設定すると、すべてのデバイスでサポートされないように強制します。1 に設定すると、すべてデバイスでサポートを強制します。
SYCL_CACHE_TRACE	任意 (*)	変数が設定されていると、キャッシュイベントまたは非ブロックエラー (キャッシュの項目にアクセスできないなど) が発生すると、std::err にメッセージが出力されます。
SYCL_PARALLEL_FOR_RANGE_ROUNDING_TRACE	任意 (*)	レンジを切り上げた parallel_for 呼び出しのトレースを有効にします。
SYCL_PI_SUPPRESS_ERROR_MESSAGE	任意 (*)	エラーメッセージの出力を抑制します (ベースとなるツールチェーンが生成するエラーを中断しないため、CI にのみ使用)。この変数は、エラーメッセージの出力 (エラー値、名前、説明、および場所) のみに影響することに注意してください。エラーコードの処理とアボート/スロー動作は変更されません。

(*) 注: 任意は、この環境変数が NULL 以外の値に設定されている場合に有効であることを意味します。

SYCL_PRINT_EXECUTION_GRAPH オプション

SYCL_PRINT_EXECUTION_GRAPH は、次の表から 1 つ以上のカンマで区切られた値を受け付けます。

オプション	説明
before_addCG	addCG メソッドの前にグラフを出力します。
after_addCG	addCG メソッドの後にグラフを出力します。
before_addCopyBack	addCopyBack メソッドの前にグラフを出力します。
after_addCopyBack	addCopyBack メソッドの後にグラフを出力します。
before_addHostAcc	addHostAccessor メソッドの前にグラフを出力します。
after_addHostAcc	addHostAccessor メソッドの後にグラフを出力します。
always	上記のメソッドの前後にグラフを出力します。

SYCL_PI_TRACE オプション

`SYCL_PI_TRACE` にはビットマスクを指定できます。サポートされるトレースレベルを以下に示します。

オプション	説明
1	PI プラグイン/デバイス検出をトレースする基本トレースを有効にします。
2	PI 呼び出しのトレースを有効にします。
-1	すべてのレベルのトレースを有効にします。

レベルゼロプラグイン向けのデバッグ変数

⚠ **警告:** 以下の環境変数は、DPC++ コンパイラーとランタイムの開発とデバッグに使用されます。これらのセマンティクスは変更される可能性があります。製品化するコードでは、これらの変数に依存しないでください。

環境変数	値	説明
`SYCL_PI_LEVEL_ZERO_SINGLE_THREAD_MODE`	整数	シングルスレッドのアプリケーションでは、このモードを有効にすると、レベルゼロプラグインでミューテックス・ロックによるオーバーヘッドを回避できます。0 より大きい値を設定すると、シングルスレッド・モードが有効になります。0 はシングルスレッド・モードを無効にします。デフォルトは 0 です。
`SYCL_PI_LEVEL_ZERO_USM_ALLOCATOR`	[EnableBuffers][;[MaxPoolSize][;[host|device|shared:][MaxPoolableSize][,[Capacity][,SlabMinSize]]]...]	`EnableBuffers` は、SYCL* バッファーのプールを有効にします。デフォルトは 0 (無効) で、1 に設定すると有効になります。`MaxPoolSize` はプールの最大サイズで、デフォルトは 0 です。`MemType` は、`host`、`device` または `shared` です。その他のパラメーターは、オプションの `K`、`M`、または `G` サフィックスが付加された正の整数値です。`MaxPoolableSize` は、プール可能な最大割り当てサイズです。デフォルトは、`host` および `shared` の場合は 0、`device` の場合は 32KB です。`Capacity` は、プログラムによって解放され、再割り当てのためプールに保持された各サイズレンジの割り当て数であり、デフォルトは 0 です。サイズレンジには次のパターンを使用します。64、96、128、192 など (2 の累乗で、その間に 1 つのレンジがあります)。`SlabMinSize` は最小割り当てサイズであり、`host` および `device` では 64KB、`shared` の場合は 2MB です。 例: `SYCL_PI_LEVEL_ZERO_USM_ALLOCATOR=1;32M;host:1M,4,64K;device:1M,4,64K;shared:0,0,2M`
`SYCL_PI_LEVEL_ZERO_BATCH_SIZE`	整数	コマンドリストを実行する前に、コマンドリストにバッチ処理する計算コマンドの推奨数を設定します。値を 0 にすると、バッチサイズは動的に調整されます。0 より大きな値では、バッチサイズは指定された値になります。デフォルトは 0 です。

環境変数	値	説明
SYCL_PI_LEVEL_ZERO_COPY_BATCH_SIZE	整数	コマンドリストを実行する前に、コマンドリストにバッチ処理するコピーコマンドの推奨数を設定します。値を 0 にすると、バッチサイズは動的に調整されます。0 より大きな値では、バッチサイズは指定された値になります。デフォルトは 0 です。
SYCL_PI_LEVEL_ZERO_FILTER_EVENT_WAIT_LIST	整数	0 に設定すると、レベルゼロ・バックエンド利用時に、待機リストからのシグナルイベントのフィルター処理が無効になります。デフォルトは 0 です。
SYCL_PI_LEVEL_ZERO_USE_COPY_ENGINE	任意 (*)	この環境変数を使用すると、ユーザーはコピー操作におけるコピーエンジンの利用を制御できます。値が整数の場合、デバイスで使用可能な場合はレベルゼロプラグインでコピーエンジンを使用して、ホストやデバイス間で SYCL* バッファーまたはイメージデータを転送し、デバイスまたは共有メモリ内の SYCL* バッファーまたはイメージデータを埋めることができます。デフォルトは 1 です。
SYCL_PI_LEVEL_ZERO_USE_COMPUTE_ENGINE	整数	整数 (>=0) を設定します。設定すると、すべての計算コマンドは、計算コマンドグループ内の指定されたインデックスを持つコマンドキューに送信されます。負の値を設定すると、利用可能なすべての計算エンジンが使用されます。デフォルト値は 0 です。
SYCL_PI_LEVEL_ZERO_USE_COPY_ENGINE_FOR_D2D_COPY (実験的)	整数	デバイスからデバイスへのコピー操作のレベルゼロプラグインで、利用可能であればコピーエンジンの利用を許可します。デフォルトは 0 です。このオプションは実験的なものであり、デバイスからデバイスへのコピー操作にコピーエンジンを使用するかどうかを決定するヒューリスティックが導入された時点で廃止されます。
SYCL_PI_LEVEL_ZERO_DEVICE_SCOPE_EVENTS	任意 (*)	ホストに状態が見えないデバイス・スコープ・イベントのサポートを有効にします。有効なモードが、SYCL_PI_LEVEL_ZERO_DEVICE_SCOPE_EVENTS=1 の場合、レベルゼロプラグインは、デバイススコープのみを持つすべてのイベントを作成し、ホストで状態 (待機/クエリー) が要求されるとプロキシー host-visible イベントを作成します。有効なモードが、SYCL_PI_LEVEL_ZERO_DEVICE_SCOPE_EVENTS=2 の場合、レベルゼロプラグインは、デバイススコープのみを持つすべてのイベントを作成し、各コマンドリスト送信の最後にプロキシー host-visible イベントを作成します。デフォルトは 0 であり、すべてのイベントがホストの可視性を持つことを意味します。SYCL_PI_LEVEL_ZERO_DEVICE_SCOPE_EVENTS は、即時コマンドリスト (SYCL_PI_LEVEL_ZERO_USE_IMMEDIATE_COMMANDLISTS = 1) を使用する場合は無視され、すべてのイベントはデフォルトスコープ 0 を使用します。

環境変数	値	説明
SYCL_PI_LEVEL_ZERO_USE_IMMEDIATE_COMMANDLISTS	整数	正の値に設定すると、レベルゼロの即時コマンドリストが利用できるようになります。これは、バッチ処理されず、すべてのコマンドが即座に送信され実行されることを意味します。1 に設定すると、SYCL* キューごとに固有の即時コマンドリストが作成されます。2 に設定すると、ホストスレッドの SYCL* キューごとに一意の即時コマンドリストが作成されます。デフォルトは 0 です。
SYCL_PI_LEVEL_ZERO_USE_MULTIPLE_COMMANDLIST_BARRIERS	整数	正の値に設定すると、バリアーを送信する際に複数のレベルゼロ・コマンド・リストを使用できるようになります。デフォルトは 1 です。
SYCL_PI_LEVEL_ZERO_USE_COPY_ENGINE_FOR_FILL	整数	正の値に設定すると、メモリーフィル操作にコピーエンジンを利用できるようになります。デフォルトは 0 です。
SYCL_PI_LEVEL_ZERO_SINGLE_ROOT_DEVICE_BUFFER_MIGRATION	整数	0 に設定すると、すべてのデバイスが同じルートを持つコンテキストで、すべてのデバイスで単一のルートデバイス割り当てを使用します。それ以外では、通常のバッファー移行を行います。デフォルトは 1 です。
SYCL_PI_LEVEL_ZERO_REUSE_DISCARDED_EVENTS	整数	正の値にすると、コマンド間の依存関係チェーンに従って破棄されたレベルゼロイベントがリセットされ、同じインオーダー・キューのレンジ内で再利用されるモードが有効になります。デフォルトは 1 です。
SYCL_PI_LEVEL_ZERO_EXPOSE_CSLICE_IN_AFFINITY_PARTITIONING (非推奨)	整数	ゼロ以外の値にすると、計算スライスが `sycl::info::partition_property::partition_by_affinity_domain` パーティション化スキームのサブのサブデバイスとして公開されます。デフォルトはゼロであり、`sycl::info::partition_property::ext_intel_partition_by_cslice` によるパーティション化が行われる場合にのみ公開されます。このオプションは互換性の目的で導入されており、今後廃止される予定です。これから開発するコードは、この動作に依存してはなりません。また、サブのサブデバイスが `partition_by_affinity_domain` によって作成された場合でも、計算スライスによるパーティション化によって作成されたとレポートされることに注意してください。
SYCL_PI_LEVEL_ZERO_COMMANDLISTS_CLEANUP_THRESHOLD	整数	負でない場合、しきい値はその値に設定されます。負の場合、しきい値は `INT_MAX` に設定されます。キュー内のコマンドリスト数がこのしきい値を超えると、以降に再利用できるよう、完了したコマンドリストのクリーンアップが試行されます。デフォルトは 20 です。
SYCL_PI_LEVEL_ZERO_IMMEDIATE_COMMANDLISTS_EVENT_CLEANUP_THRESHOLD	整数	負でない場合、しきい値はその値に設定されます。負の場合、しきい値は `INT_MAX` に設定されます。即時コマンドリストに関連付けられたイベント数がこのしきい値を超えると、通知されたイベントがチェックされ、それらのイベントは再利用されます。このしきい値を低くすると、イベントが頻繁にチェックされるようになり、不要なイベントが早期に再利用される可能性があります。ただし、イベント状態チェックを頻繁に行うと時間がかかります。デフォルトは 20 です。

環境変数	値	説明
SYCL_PI_LEVEL_ZERO_USM_RESIDENT	整数	割り当て時に USM 割り当てを常駐にするかどうか、またその場所を制御します。0 (デフォルト) に設定すると、特別な常駐は強制されません。1 に設定すると、割り当て (デバイスまたは共有) は割り当てデバイスに常駐します。2 に設定すると、割り当て (デバイスまたは共有) は、割り当てデバイスに P2P アクセスできる、割り当てコンテキスト内のすべてのデバイスに常駐します。ホスト割り当ての場合、0 以外の設定はコンテキスト内のすべてのデバイスに割り当ての常駐を強制します。
SYCL_PI_LEVEL_ZERO_USE_NATIVE_USM_MEMCPY2D	整数	正の値に設定すると、レベルゼロ USM 2D メモリーのコピー操作が有効になります。デフォルトは 0 です。

CUDA* プラグイン向けのデバッグ変数

⚠ **警告:** 以下の環境変数は、DPC++ コンパイラーとランタイムの開発とデバッグに使用されます。これらのセマンティクスは変更される可能性があります。製品化するコードでは、これらの変数に依存しないでください。

環境変数	値	説明
SYCL_PI_CUDA_ENABLE_IMAGE_SUPPORT (実験的)	任意 (*)	イメージのサポートを有効にします。イメージのサポートはまだ完全に実装されていないため、このオプションは実験的なものです。

(*) 注: 任意は、この環境変数が NULL 以外の値に設定されている場合に有効であることを意味します。