APPENDIX A

XML の詳細規則

この付録は、次の内容で構成されています。

（注） 詳細については、「[Advanced Rules]」（P.9-17）を参照してください。

Cisco NAC Profiler の詳細規則オプションにより、Extensible Markup Language（XML）を使用してカスタム ルールを定義し、ブール論理演算子 OR、AND、および NOT を使用して Profiler ルール タイプを組み合わせることができます。単独のタイプの複数のルールと異なるタイプのルールを組み合わせて 1 つの詳細規則にまとめ、NetMap、NetWatch、および NetRelay（トラフィック ルール用）の各モジュールを経由して Cisco NAC Profiler が収集したエンドポイント データを評価できます。

この付録は、「[Advanced Rules]」（P.9-17）で提供されている情報をさらに掘り下げて、上級 Profiler 管理者が製品の拡張機能である詳細規則の設計と設定を行うことを支援します。適切に設計された場合、詳細規則により Cisco NAC Profiler の柔軟性が著しく向上します。ただし、詳細規則に無効な XML 設定があると Profiler Server の Server モジュールが起動できなくなり、不適切に構築されたルールのロジックが、システム全体のパフォーマンスに悪影響を与える場合があります。現在のバージョンでは、設定にルールを保存する前に、詳細規則エディタは詳細規則に無効な XML がないかをチェックします。これである程度のエラー チェックができます。

どの高度な技術でも同じですが、Cisco NAC Profiler 製品固有のマニュアルと要件に必要なレベルの注意を払って詳細規則に取り組むことをお勧めします。たとえば、Profiler は、侵入防止（IPS）などの他のアプリケーションで使用されるものと同様の技術を活用していますが、Cisco NAC Profiler は IPS としては設計されていません。これらの上級テクニックを使用してプロファイルを構成する場合や Cisco NAC Profiler の導入を計画する場合には、この点を考慮することが重要です。詳細規則の設計に関して疑問がある場合、またはパフォーマンスに影響が生じると思われる場合は、Cisco NAC Profiler のテクニカル サポートに連絡して指示を仰ぐのが適切です。

XML 詳細規則の構文

詳細規則は、「Cisco NAC Profiler ドキュメント タイプ定義（DTD）」（P.A-9）に記述されている Profiler Document Type Description（DTD）に準拠する有効な XML 構文を使用して設定されます。詳細規則の一般的な形式は次のようになります。

```
<Rule name="Advanced Rule">
```

```
  <RuleEntity entity="Profile Name" cf="certainty_value"/>
  <logical_operator 1>
    Rule 1
    Rule n
  </logical_operator 1>
</Rule>
```

このドキュメントでは、論理ブロックを論理演算子 AND、OR、または NOT で囲まれた 1 つまたは複数のルールとして定義しています。複数の論理ブロックを論理演算子内で「入れ子」にして、複数の基準をチェックする複合ルールを作成できます。NOT 演算子は、他の基準を除外するのに設計できます。

詳細規則では、Profiler Modeling Engine（Modeler）は、最初の行でルール名構文を使用しません。これは、ルールを識別できるようルールに名前を付けるためだけに使用されます。ルールには任意の名前を付けることができますが、詳細規則を作成するときは、システムを簡単に説明するわかりやすい名前を付けることを強くお勧めします。

詳細規則の 2 番目の行の RuleEntity は重要です。この行により、Modeler は各詳細規則をバインドするプロファイルに関連付けます。

各詳細規則の 2 番目の行では、次の構文を使用する必要があります。

```
<RuleEntity entity="[Profile-Name]" cf="x"/>
```

上記で

- **[Profile-Name]** はプロファイル名（大文字と小文字が区別されます）と置き換えられます。
- **x** はこのルールの確実度値である .01 から 1 の間の値に置き換えられます。

次の例では、名前付き詳細規則が Advanced Printer という名前のプロファイルにバインドされると解釈されます。この詳細規則の論理が真であると判定されるエンドポイントは、65% の確実度で Advanced Printer プロファイルに配置されます。

```
<RuleEntity entity="Advanced Printer" cf="0.65"/>
```

この後で説明するように、詳細規則の「entity=」行の直後に、ルール ロジックを定義するルールの構文が適切な XML 形式で入力されます。

（注） ルールの元の作成者の助けを借りずに、他者がルールの基礎となる論理を理解できるよう、詳細規則にコメントを挿入して適切なドキュメントを作成することをお勧めします。次の構文を使用して、詳細規則にコメント行を追加できます。

```
<!-- comment text -->
```

複数の論理ブロック

詳細規則を使用する主なメリットとして、ルール ロジック内で AND や NOT のような他の利用可能な論理演算子を使用できたり、MAC Vendor および DHCP Vendor Class のような異なるルール タイプを 1 つのルール内で使用できることがあります。

Profiler Server の GUI プロファイル エディタを通じて複数のルールがプロファイルに追加されると、個別のルールが独立したエンティティとして Modeler に追加されます。リモデリング中は、Modeler は有効なプロファイルに指定された各ルールを、標準プロファイル作成プロセスを通じて個別に評価する必要があります。GUI を使用して複数のルールがプロファイルに追加される場合、デフォルト論理演算子は OR です。プロファイルの個別のルールは独立して評価されるため、複数のルールは真である

と判定できます。したがって、複数ルールが一致した結果生じる確実度の組み合わせを表す確実度アルゴリズムが使用されます。複数ルールをプロファイルで定義する GUI の使用に関して考慮すべき重要な概念は、それらが Modeler によって独立して評価されるという点です。

対照的に、詳細規則では論理演算子を使用した複数の論理ブロックが可能で、これらの論理ブロックをさらに演算子内で入れ子にして、最大限の柔軟性を実現します。ただし、詳細規則の場合は、ルールに埋め込まれたロジックが、指定した論理演算にしたがって順次評価されます。たとえば、AND 演算子内で 2 つ以上の論理ブロックが入れ子になっており、最初のブロックが偽であると判定された場合、AND ではすべてのブロックが真と判定される必要があるため、Modeler はルール中の残りの論理ブロックの評価を中止します。同様に、OR 演算子内で 2 つ以上の論理ブロックが入れ子になっている場合、1 つの論理ブロックが真であると判定されると、詳細規則の条件がただちに満たされ（つまり少なくとも 1 つの条件が真であると判定される）、Modeler は詳細規則中の残りの論理ブロックの評価を中止します。

したがって、複数の基準で判定を行う場合は、GUI を使用して複数ルールを指定するよりもはるかに効率的な、詳細規則（適切に構築されている場合）を定義できます。

OR 演算子とルールおよび論理ブロックの順番

詳細規則ロジックを作成するときは、プレフィクス表記法が使用されます。プレフィクス表記法では、ロジックを適用する項目（ルール）の外側に論理演算子（例：AND、OR、NOT）を指定します。たとえば、汎用プリンタ プロファイルのシステム生成詳細規則から取得された次の論理ブロックを考えてみます。

```
<OR>
  <TrafficRule ipaddr="0.0.0.0" sport="0" dport="515"/>
  <TrafficRule ipaddr="0.0.0.0" sport="0" dport="9100"/>
</OR>
```

上記の論理ブロックの例では、OR 演算子の内部、つまり <OR> と </OR> の間に 2 つのトラフィック ルールが定義されています。このルールを評価するときは、Modeler は最初のルールを評価して、次に必要な場合にだけルール自体によって示される順番で 2 番目のルールを評価します。論理演算子が OR であるため、この論理ブロックは*いずれか*のトラフィック ルールが真である場合だけ真と判定されます。したがって、リストの最初のルールが真であると判定された場合、Modeler は残りのルールのチェックを続けません。この論理ブロックは、Target of Evaluation（TOE）に対して真であると判定され、Modeler は次のブロックまたは次のルールの評価を続けます。

OR 演算子を使用して論理ブロック内で複数のルールが指定されている場合は、真であると判定される可能性が最も高いルールがリストの先頭の論理ブロックに定義されるよう、論理ブロック内のルールに順番を付けるのが適切な方法です。同様に、論理ブロックが OR 内で入れ子になっている場合でも、Modeler ができるだけ効率的に詳細規則をヒットして終了できるよう、真であると判定される可能性が最も高いルールを最初に配置する必要があります。

これは、詳細規則の効率的な設計における重要な一般的概念を浮き彫りにしています。AND 演算子ではすべてのルール（または入れ子にされた論理ブロック）が真であると判定されなければならないため、AND を戦略的に使用するとルールがさらに効率的になります。AND 論理演算子を詳細規則内で使用して、ルールの「プレ修飾子」である論理ブロックを作成して、Modeler の効率性とパフォーマンスを向上させることができます。

汎用プリンタ ルール

この原則が実際に使用できる様子を次の例で説明します。汎用プリンタ ルールは、MyNetwork で指定されたプリント サーバから、標準のプリンタ ポートであるポート 9100 から 515 までのエンドポイントのトラフィックに一致する一連のトラフィック ルールを含む詳細規則です。このシステム生成され

た詳細規則におけるプライマリ論理ブロックは、次の例に示すように、それぞれの既知のプリント サーバ IP アドレス（MyNetwork 設定に追加されている）の 2 つのトラフィック ルール上の OR 演算子を使用して作成されます。

```
<OR>
 <TrafficRule ipaddr="1.1.2.50" sport="0" dport="515"/>
 <TrafficRule ipaddr="1.1.2.50" sport="0" dport="9100"/>
 <TrafficRule ipaddr="1.1.2.51" sport="0" dport="515"/>
 <TrafficRule ipaddr="1.1.2.51" sport="0" dport="9100"/>
 <TrafficRule ipaddr="1.1.2.52" sport="0" dport="515"/>
 <TrafficRule ipaddr="1.1.2.52" sport="0" dport="9100"/>
 <TrafficRule ipaddr="1.1.2.53" sport="0" dport="515"/>
 <TrafficRule ipaddr="1.1.2.53" sport="0" dport="9100"/>
</OR>
```

この例では、次の 4 つのプリント サーバのアドレスがわかっており、Cisco NAC Profiler の MyNetwork 設定に入力されています。

```
1.1.2.50
1.1.2.51
1.1.2.52
1.1.2.53
```

上記のとおり、論理ブロックが自分で使用されている場合は、一致が見つかるまで Modeler が各トラフィック ルールを評価します。最初の一致で、Target of Evaluation（ToE）について詳細規則が真であると判定された時点で Modeler が終了します。

AND 演算子

特定の論理ブロック（「汎用プリンタ ルール」（P.A-3）を参照）を使用した AND 演算子、さらに次の例に示すような新しいより一般的な「プレスクリーニング」論理ブロックを使用して、詳細規則の効率性を高めることができます。この例では、システム生成された汎用プリンタ用の詳細規則を、現行の Profiler バージョンで実際に構築する方法を示します。

プレスクリーン AND 論理ブロック

```
<AND>
        <OR>
         <TrafficRule ipaddr="0.0.0.0" sport="0" dport="515"/>
         <TrafficRule ipaddr="0.0.0.0" sport="0" dport="9100"/>
        </OR>
        <OR>
         <TrafficRule ipaddr="1.1.2.50" sport="0" dport="515"/>
         <TrafficRule ipaddr="1.1.2.50" sport="0" dport="9100"/>
         <TrafficRule ipaddr="1.1.2.51" sport="0" dport="515"/>
         <TrafficRule ipaddr="1.1.2.51" sport="0" dport="9100"/>
         <TrafficRule ipaddr="1.1.2.52" sport="0" dport="515"/>
         <TrafficRule ipaddr="1.1.2.52" sport="0" dport="9100"/>
         <TrafficRule ipaddr="1.1.2.53" sport="0" dport="515"/>
         <TrafficRule ipaddr="1.1.2.53" sport="0" dport="9100"/>
        </OR>
       </AND>
```

AND 演算子と新しい OR 論理ブロックが、「汎用プリンタ ルール」（P.A-3）に示すような既知のプリント サーバからのトラフィックに具体的に一致する元の論理ブロックに追加されます。

新しい論理ブロックと AND 演算子を共に使用することにより、トラフィックがプリント サーバ（つまり元の OR 論理ブロック）の指定されたリストから送信されるのか評価する前に、印刷に関連付けられたいずれかのポート番号からトラフィックが送信されるようになります。AND 演算子と共に使用した場合、この方法は詳細規則のプレスクリーニング機能を実現します。どちらの論理ブロックも一般的な AND 固有の性格を持ち、条件を満たすには真であると判定される必要があります。一般的な論理ブロックが偽であると判定された（つまり、トラフィックが印刷に関連付けられた既知のポート番号に存在しない）場合、Modeler はルールを終了し、サイクルを浪費することなく先に進んで、これが指定されたポートの固有の IP アドレスからのトラフィックであるかどうかを検証します。この方法により、特に、特定の論理ブロックに数百台のプリント サーバといったような大量のエントリが存在するような状況でパフォーマンスが大きく向上します。プレスクリーニング ブロックを使用してルールを設計することで、Modeler は印刷に関連付けられていることがわかっているトラフィックだけを評価し、基準に一致しないトラフィックを評価しなくてもすみます。

NOT 演算子

詳細規則での NOT 演算子の使用方法は他の論理演算子の場合と似ており、論理ブロックで NOT 演算子を AND 演算子または OR 演算子と結合するのが一般的です。NOT は、共通の属性を共有するエンドポイントから特定の基準に一致するエンドポイントを除外するのに使用できます。

たとえば、詳細規則で、指定したサブネット上のすべてのエンドポイントに一致して、.1 から .10 までのホスト アドレス（最後のオクテット）とサブネット ブロードキャスト（.255）アドレスを持つデバイスを除外する必要がある場合、NOT 演算子を使用する次の論理ブロックを使用できます。

```
<AND>
    <AddressRule ipaddr="10.180.0.0" mask="255.255.255.0"/>
  <NOT>
     <OR>
       <AddressRule ipaddr="10.180.0.0" mask="255.255.255.248"/>
       <AddressRule ipaddr="10.180.0.8" mask="255.255.255.255"/>
       <AddressRule ipaddr="10.180.0.9" mask="255.255.255.255"/>
       <AddressRule ipaddr="10.180.0.10" mask="255.255.255.255"/>
       <AddressRule ipaddr="10.180.0.255" mask="255.255.255.255"/>
    </OR>
   </NOT>
  </AND>
```

太字のセクション NOT 演算子は入れ子にされており、OR 演算子と特定のアドレスに一致するルールを使用して論理ブロックをこのルールに一致するものから除外しています。この場合も、AND 演算子内のどちらの論理ブロックも、Modeler が ToE をこのルールに一致させるため真であると判定する必要があります。最初の論理ブロックはサブネット 10.180.0 上のすべてのホスト アドレスに一致し、.1 以外で .10 から .255 までのホストアドレスとして解釈される論理ブロックを否定します。

次の例は、既知の Novell Server と通信を行う場合に、（アドレス ルールを使用したホスト IP アドレスを使用して）ワイヤレスで接続されるものでないことがわかっている特定のエンドポイントを除外するため、詳細規則で NOT 演算子を使用する方法を示しています。この例では、ワイヤレスで接続されているエンドポイントには、サブネット 10.200.0.0/21 上にホスト アドレスが割り当てられていることがわかっています。したがって、NOT 演算子は、（トラフィック ルールなどを使用して）指定したポート番号の指定した Novell Server との通信を示すトラフィック ルールに一致するホストから、このサブネット上のアドレスを持つホストを除外するのに使用されます。

```
<Rule name="Novell Employee">
  <RuleEntity entity="Novell_Employee" cf="0.90"/>
  <AND>
```

```
    <NOT>
       <AddressRule ipaddr="10.200.0.0" mask="255.255.248.0"/>
    </NOT>
    <TrafficRule ipaddr="192.168.1.101" dport="524" sport="0"/>
  </AND>
</Rule>
```

詳細規則内での REGEX の使用

詳細規則の論理ブロック内で使用されるルール タイプの多くは、GUI を通じて定義される標準ルールのように、データの一致に正規表現（REGEX）を利用します。このドキュメントの多くの例では、論理ブロック内で REGEX が使用されています。

詳細規則内でのトラフィック ルールと Web URL ルールの使用

第 9 章「エンドポイント プロファイルの設定」で説明しているように、トラフィック ルールと Web URL アプリケーション ルール タイプでは、特定のネットワーク トラフィック、つまりプロファイル ルール自体で指定されている特定の基準を満たすトラフィックを Cisco NAC Profiler が収集する必要があります。これには、NetWatch モジュールと NetRelay（トラフィック ルールだけの場合）モジュール両方を設定し、これらのルール タイプを利用する有効なプロファイルに一致するルールについて Modeler が評価するトラフィックを収集する必要があります。

GUI を使用して、プロファイル内にトラフィック ルールおよび Web URL アプリケーション ルールを作成する場合は、Collector の監視ポートから、または NetRelay を実行する Collector の NetFlow を通じて、Profiler がこのネットワーク トラフィックの収集を開始するのに必要な NetWatch モジュールと NetRelay モジュールへの設定変更が、システム全体に自動的に行われます。

トラフィック ルールまたは Web URL アプリケーション ルールを使用したプロファイルの追加または有効化の後の [Apply Changes] -> [Update Modules] の間、システムのすべての NetWatch モジュールと NetRelay モジュールの新しい XML 設定ファイルが生成され、モジュールが再起動されると、Modeler がルールに一致するデータの検証を開始できるよう、このデータの収集が開始されます。

（注） URL ルールの場合は NetRelay の設定は変更されず、NetWatch の設定だけ変更されます。トラフィック ルールでは、特定の収集について両方のモジュールを設定する必要があります。

ただし、バージョン 2.1.8 以前の GUI では XML 詳細規則を解析しません。したがって、トラフィック ルールまたは Web URL アプリケーション ルールを詳細規則で使用する場合は、これらのルールに対して評価するため特定のネットワーク トラフィックの収集に関する NetWatch モジュールと NetRelay モジュールの設定は自動的に行われず、手動で設定する必要があります。これは、単に GUI を使用してルールを追加することで実現できます。標準トラフィック ルールまたは Web URL ルールが、トラフィック ルールまたは Web URL ルールを含む詳細規則を持つプロファイルに追加されます。「収集ルール」として機能する標準ルールが、低いレベルの確実度（通常は 1% で十分）で追加されます。Profiler が NetWatch または NetRelay を通じてトラフィックの収集を開始し、詳細規則に対して評価を行うために必要な特定の収集を有効にすることが目的です。

特に、トラフィック ルール向けのこれらの「収集ルール」の構造は、このドキュメントの前半で説明したように、パフォーマンスへの影響があり、GUI を使用して標準ルールを追加するたびに、Modeler が評価する必要があります。したがって、これらの収集ルールをオープンにし、特定の宛先アドレスまたは発信元アドレスの標準ルールを追加しようとしない方法が採用できます。つまり、該当する収集

ルールで、発信元または宛先アドレスの IP アドレスに 0.0.0.0 を使用することです。これにより、指定した発信元ポート番号または宛先ポート番号で検知されるすべてのトラフィックが収集されます。詳細規則ロジックを使用して目的の一致を識別します。識別された特定のホストを持つ収集ルールのリストを設定しないようにし、Modeler に各収集ルールと詳細規則を評価させるのが適切な方法です。

この概念を説明するために、「NOT 演算子」（P.A-5）で説明した詳細規則について検討します。これには、ポート 524 の Novell Server IP と通信するエンドポイントに一致するよう設計されたトラフィック ルールが含まれています。同じプロファイル（例にあるプロファイル名は Novell_Employee）で、GUI を使用して標準トラフィック ルールが定義されており、詳細規則の収集ルールの役割を果たしています。そのトラフィック ルールには次のパラメータがあり、トラフィックの収集のために NetWatch モジュールと NetRelay モジュールがセットアップされています。

```
0.0.0.0
Dest IP
Src Port 0
Dest Port 524
Certainty 1%
```

収集ルールとして使用される標準トラフィック ルールと詳細規則を Novell_Employee プロファイルに追加し、プロファイルを有効にしたら、[Apply Changes] -> [Update Modules] を実行して NetWatch と NetRelay の設定を生成し、詳細規則と一致する必要のあるトラフィックの収集を開始する必要があります。再起動後、ポート 524 の Novell Server と通信するがワイヤレス ホスト アドレスを持たないという基準に一致するエンドポイントが検知されると、それらのエンドポイントは Novell_Employee プロファイルに 90% の確実度（上記の詳細規則で指定）で追加されます。

収集ルールと詳細規則を Novell_Employee プロファイルに追加するという手順を実施しない場合は、NetWatch と NetRelay はこのトラフィックを収集するようプログラムされず、一致は起こりません。

詳細規則の例

このセクションで説明している詳細規則の例は、詳細規則構造の例を追加し、このセクションの概念をさらに詳しく説明するよう作成されています。

（注）　このセクションのルールは、適切に構築されていても、すべての環境でそのまま使用できるわけではありません。これらのルールは、主に詳細規則で利用可能なルール タイプとロジックをプロファイルの特定のエンドポイントに結合する方法を示す例として提供されています。

複数の論理ブロックを使用した AND 演算子

この最初の例では、複数の論理ブロックを使用した AND 演算子を使用しており、1 つは単独の MAC ベンダー ルールで構成されており、もう一方は OR 演算子を適用した複数の論理ブロックで構成されています。このルールに一致するデバイスは、OUI に Cisco（大文字と小文字が区別されます）を含む MAC アドレス、および 192.168.240.0 ネットワークの指定された 17 のサブネットのうちの 1 つの IP アドレスを持っています。

```
<Rule name="CiscoRouterAdv">
      <RuleEntity entity="Cisco Infrastructure" cf="0.6"/>
    <AND>
      <Vendor vendor="/^cisco/i"/>
     <OR>
      <AddressRule ipaddr="192.168.208.0" mask="255.255.255.240"/>
      <AddressRule ipaddr="192.168.208.16" mask="255.255.255.240"/>
```

```
    <AddressRule ipaddr="192.168.208.48" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.64" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.80" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.96" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.112" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.128" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.144" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.160" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.176" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.192" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.208" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.224" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.208.240" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.209.0" mask="255.255.255.240"/>
    <AddressRule ipaddr="192.168.209.16" mask="255.255.255.240"/>
   </OR>
  </AND>
 </Rule>
```

アクセス ポイントと DHCP ベンダー クラス

次の例は、次のアクセス ポイントのプロファイルとなる詳細規則です。

- ポート 5000（Enterasys/Trapeze AP のデフォルト トンネル ポート）で通信を行う IP アドレス 10.4.1.5 の中央ワイヤレス スイッチとの通信を行うアクセス ポイント。
- Profiler によって検知され、DHCP ベンダー クラス WIRELESS-AP:AP1002、WIRELESS-AP:AP3000/I、または WIRELESS-AP:AP4102 のいずれかを持つ DHCP 要求を送信するアクセス ポイント。

```
<Rule name="Advanced Access Points">
      <RuleEntity entity="ETS Access Points" cf="0.90"/>
    <AND>
      <TrafficRule ipaddr="10.4.1.5" sport="0" dport="5000"/>
     <OR>
      <DHCPVendor vendor="/WIRELESS-AP:AP1002/i"/>
      <DHCPVendor vendor="/WIRELESS-AP:AP3000/i"/>
      <DHCPVendor vendor="/WIRELESS-AP:AP4102/i"/>
     </OR>
    </AND>
   </Rule>
```

Wireless Windows エンドポイント

この例では、ワイヤレス サブネット上のホスト アドレスに基づき、ワイヤレスで接続されることがわかっている Windows バージョンを実行するエンドポイントを識別します。WebClient/Web ユーザ エージェント ルールを使用して、Windows OS に関連付けられた Web クライアントを表示するデバイスを識別します。このルールは、論理ブロック内のルールの順序に関してすでに説明した原則の 1 つを示しており、ワイヤレス ネットワーク サブネットは、早期に一致が起こり論理ブロックを終了できるように、可能性の高いものから可能性の低いものへと並べられます。

```
<Rule name="Win User Wireless">
      <RuleEntity entity="Windows Users - Wireless" cf="0.85"/>
    <AND>
     <OR>
      <AddressRule ipaddr="10.1.0.0" mask="255.255.255.0"/>
      <AddressRule ipaddr="10.1.1.0" mask="255.255.255.0"/>
```

```
  <AddressRule ipaddr="10.1.2.0" mask="255.255.255.0"/>
  <AddressRule ipaddr="10.1.4.0" mask="255.255.252.0"/>
  <AddressRule ipaddr="10.2.0.0" mask="255.255.255.0"/>
  <AddressRule ipaddr="10.2.1.0" mask="255.255.255.0"/>
  <AddressRule ipaddr="10.2.2.0" mask="255.255.255.0"/>
  <AddressRule ipaddr="10.2.4.0" mask="255.255.252.0"/>
 </OR>
 <OR>
  <WebClient banner="/Windows|Win32/i"/>
  <DHCPVendor vendor="MSFT 5.0"/>
  <DHCPVendor vendor="MSFT 98"/>
  <DHCPVendor vendor="Microsoft Windows CE"/>
 </OR>
</AND>
</Rule>
```

Cisco NAC Profiler ドキュメント タイプ定義（DTD）

XML ドキュメントの有効な構築ブロックを定義することが DTD の目的です。有効な要素と属性のリストを使用してドキュメント構造を定義します。Cisco NAC Profiler ルールとイベントの DTD は、「ルールとイベントの Profiler DTD」（P.A-10）に参考として提示されています。

現行バージョンの Cisco NAC Profiler では、管理者が定義する詳細規則は DTD に対して検証されません。ただし、バージョン 2.1.8 では XML パーサーが追加され、詳細規則 エディタで基本的な XML 構文をチェックします。

警告　**詳細規則で不適切な XML 構文を使用し、システム設定に追加されると、[Apply Changes] -> [Update Modules] によるシステムの再起動時に、システムのサーバ モジュールで障害が発生します。**

XML 詳細規則を設定に保存する前に構文がチェックされ、詳細規則が有効な XML 形式でない場合は、エラーが表示されます。XML 詳細規則を作成して XML の有効性と DTD への準拠を確認する場合、このドキュメントの例と 「ルールとイベントの Profiler DTD」（P.A-10）の DTD を使用することを強くお勧めします。

Profiler の DTD を確認することで、XML 詳細規則オプションで利用可能なさまざまなタイプのルールと、提供する必要のあるデータを確認できます。このドキュメントの前半で説明した論理演算子を使用する他に、現行リリースの Profiler の GUI では使用できないものの、詳細規則で使用できるルール タイプがいくつかあります。その詳細規則でだけ利用可能なルールの例には、次のものがあります。

- 要求された、または割り当てられた DHCP オプションに基づいてルールを定義できる DHCP オプション
- TTL、Window サイズ、TCP オプションなどエンドポイントのいくつかのネットワーク スタック情報属性に基づいてルールを定義できる StackRule

（注） TTL 値に関する重要な警告：NetWatch は、値が 8 で割れない場合にエンドポイントのネットワーク トラフィックで検知される TTL 値を自動的に調整します。エンドポイントからのトラフィックが、モニタ インターフェイスに受信される前に 1 つまたは複数のルータを横断する場合、このメカニズムにより TTL は、エンドポイント上のスタックで設定された元の値にリセットされます。この設計は、複数の値を使用してルールを作成（例:<OR><... ttl="32" .../><... ttl="31"...>...</OR> など）しなくてもすむようにすることを目的としています。TTL 標準の中には必ずしも 2 乗にならないものがあるため、8 が選択されています。48 と 60 を使用する旧型システムもあります。パケットで検知される TTL 値 >= 256 は、自動的に 255 にリセットされます。

これらの 2 つのルール タイプは、次の例に示す詳細 XML ルールでだけ使用できます。ルールの 1 つは要求された DHCP オプションを通じて Apple マシンを検出（MAC OS は、DHCP 要求で DHCP ベンダー クラスとして認識されない）し、もう 1 つは Microsoft Windows バージョン 3.1 を実行するエンドポイントを検出します。

```
<Rule name="MacOSAdv">
       <RuleEntity entity="Mac OS" cf="0.75"/>
       <AND>
        <Vendor vendor="/^Apple/i"/>
        <DHCPReqOptions option-list="/^1,3,6,15,112,113,78,79,95/"/>
       </AND>
    </Rule>


<Rule name="Win31Adv">
       <RuleEntity entity="Windows 3.1" cf="0.50"/>
       <AND>
        <StackRule list="/^2$/" ttl="32" window="8192" df="2"/>
        <DHCPReqOptions option-list="/^1,3,15,6,44,46,47,43/"/>
       </AND>
    </Rule>
```

ルールとイベントの Profiler DTD

このセクションでは、参考に Cisco NAC Profiler ルールの DTD とイベントを説明します。

```
<!ELEMENT ModelConfiguration ( Entity+,
                               Rule+,
                               Event*,
                               CCARule* ) >

<!ELEMENT Entity EMPTY >
<!ATTLIST Entity name        CDATA #required
                 description CDATA #implied
                 user        ( t | f )  "f"  >

<!ELEMENT Rule ( RuleEntity,
                 AddressRule*,
                 Vendor*,
                 DHCPHost*,
                 DHCPVendor*,
                 DHCPReqOptions*,
                 DHCPOptions*,
                 WebServer*,
                 WebClient*,
```

```
                    SMTPServer*,
                    DNS*,
                    SNMP*,
                    CTNBT*,
                    FTPServer*,
                    SSHServer*,
                    WebURL*,
                    StackRule*,
                    TrafficRule*,
                    PortRule*,
                    AND*,
                    OR*,
                    NOT* ) >
<!ATTLIST Rule name CDATA #implied >

<!ELEMENT AND  ( AddressRule*,
                  Vendor*,
                  DHCPHost*,
                  DHCPVendor*,
                  DHCPReqOptions*,
                  DHCPOptions*,
                  WebServer*,
                  WebClient*,
                  SMTPServer*,
                  DNS*,
                  SNMP*,
                  CTNBT*,
                  FTPServer*,
                  SSHServer*,
                  WebURL*,
                  StackRule*,
                  TrafficRule*,
                  PortRule* ) >

<!ELEMENT OR   ( AddressRule*,
                  Vendor*,
                  DHCPHost*,
                  DHCPVendor*,
                  DHCPReqOptions*,
                  DHCPOptions*,
                  WebServer*,
                  WebClient*,
                  SMTPServer*,
                  DNS*,
                  SNMP*,
                  CTNBT*,
                  FTPServer*,
                  SSHServer*,
                  WebURL*,
                  StackRule*,
                  TrafficRule*,
                  PortRule* ) >

<!ELEMENT NOT  ( AddressRule*,
                  Vendor*,
                  DHCPHost*,
                  DHCPVendor*,
                  DHCPReqOptions*,
                  DHCPOptions*,
                  WebServer*,
                  WebClient*,
                  SMTPServer*,
                  DNS*,
                  SNMP*,
```

```
                    CTNBT*,
                    FTPServer*,
                    SSHServer*,
                    WebURL*,
                    StackRule*,
                    TrafficRule*,
                    PortRule* ) >

<!ELEMENT RuleEntity EMPTY>
<!ATTLIST RuleEntity name CDATA #required
                     cf   CDATA #required>

<!ELEMENT AddressRule EMPTY>
<!ATTLIST AddressRule ipaddr CDATA #required
                      mask   CDATA #required >

<!ELEMENT Vendor EMPTY>
<!ATTLIST Vendor vendor CDATA #required >

<!ELEMENT DHCPHost EMPTY>
<!ATTLIST DHCPHost name CDATA #required >

<!ELEMENT DHCPVendor EMPTY>
<!ATTLIST DHCPVendor vendor CDATA #required >

<!ELEMENT DHCPReqOptions EMPTY>
<!ATTLIST DHCPReqOptions option-list CDATA #required >

<!ELEMENT DHCPOptions EMPTY>
<!ATTLIST DHCPOptions option-list CDATA #required >

<!ELEMENT WebServer EMPTY>
<!ATTLIST WebServer banner CDATA #required >

<!ELEMENT WebClient EMPTY>
<!ATTLIST WebClient banner CDATA #required >

<!ELEMENT SMTPServer EMPTY>
<!ATTLIST SMTPServer banner CDATA #required >

<!ELEMENT DNS EMPTY>
<!ATTLIST DNS name CDATA #required >

<!ELEMENT SNMP EMPTY>
<!ATTLIST SNMP description CDATA #required >

<!ELEMENT CTNBT EMPTY>
<!ATTLIST CTNBT nbt CDATA #required >

<!ELEMENT FTPServer EMPTY>
<!ATTLIST FTPServer banner CDATA #required >

<!ELEMENT SSHServer EMPTY>
<!ATTLIST SSHServer banner CDATA #required >

<!ELEMENT WebURL EMPTY>
<!ATTLIST WebURL url CDATA #required >

<!ELEMENT StackRule EMPTY>
<!ATTLIST StackRule ttl     CDATA #implied
                    df      CDATA #implied
                    scale   CDATA #implied
                    mss     CDATA #implied
                    window  CDATA #implied
```

```
                    list    CDATA #implied >

<!ELEMENT TrafficRule EMPTY>
<!ATTLIST TrafficRule ipaddr CDATA #required
                      sport  CDATA #required
                      dport  CDATA #required >

<!ELEMENT PortRule EMPTY>
<!ATTLIST PortRule port CDATA #required >

<!-- Events -->
<!ELEMENT Event EMPTY>
<!ATTLIST Event name   CDATA #required
                valid  CDATA #required
                id     CDATA #required
                level  CDATA #required
                fauth  ( t | f ) "f"
                snmp   ( t | f ) "f"
                web    ( t | f ) "f"
                syslog ( t | f ) "f"
                asm    ( t | f ) "f" >

<!-- CCA Rules -->
<!ELEMENT CCARule EMPTY>
<!ATTLIST CCARule name   CDATA #required
                  match  CDATA #required
                  cf     CDATA #required
                  type   CDATA #required
                  role   CDATA #required >
```